Títol: Interfície Web per execució remota de Scheduling per Grids Computacionals

Volum: 1 de 1
Alumne: Javier Miranda Jiménez

Director/Ponent: Fatos Xhafa

Departament: LSI

Data: 13 de Junio de 2007

DADES DEL PROJECTE

Títol del Projecte:

Nom de l'estudiant:
Titulació:
Crèdits:
Director/Ponent:
Departament:

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President:

Vocal:

Secretari:

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Índice

1. Introducción

Los “Computationals Grids”, o Mallas Computacionales, son un nuevo paradigma de la programación distribuida de gran escala. Los Grids son los encargados de unificar los recursos computacionales de máquinas distribuidas geográficamente en una sola máquina virtual.

Sin embargo, resulta complicado hacer estudios experimentales de aplicaciones o algoritmos sobre este nuevo paradigma, debido al gran número de factores que determinan el escenario de ejecución y la dificultad de controlar los cambios constantes de éstos.

Por ejemplo en un Grid computacional puede darse el caso de que no funcionen todas las máquinas que lo forman, o que algunas de las máquinas tengan una carga muy alta de trabajo.

Debido a estas dificultades la simulación de estos entornos juega un papel importante para poder investigar y desarrollar aplicaciones.

1.1. Conceptos previos

A continuación se explican algunos conceptos previos para facilitar la lectura de este PFC. En concreto se explicará qué es un Grid Computacional, qué son los Schedulers y en qué se basa nuestro modelo de simulación.

1.1.1. Computacionals Grids

Se adopta el término de Computacional Grid para designar una infraestructura distribuida de recursos computacionales hardware, altamente heterogéneos (en cuando a su poder de cálculo o su arquitectura), interconectados por redes de comunicación heterogéneas, y por un middleware, que ofrecen acceso de forma global, eficiente, transparente, sencilla, fiable y de bajo coste a su potencial de computación.

La primera idea que nos puede surgir al leer el párrafo anterior es si realmente necesitamos una tecnología de semejantes características. La respuesta la tenemos a continuación:

Los ordenadores son muy frecuentemente usados para modelizar y simular problemas complejos, que desafían o exceden nuestra habilidad para resolverlos, típicamente porque requieren procesar una gran cantidad de operaciones o de datos. Por otro lado, los ordenadores siempre han ido incrementando su capacidad de calculo de forma exponencial, consiguiendo de esta manera que obtengamos los resultados de todo lo que habíamos deseado.

El problema llega cuando nuestro saber no queda satisfecho con lo que nos puede proporcional un computador, necesitamos saber más de lo que es capaz de procesar un ordenador. Un ejemplo sería el caso de los biólogos quienes hace unos años se centraban en conocer la estructura de una única molécula, hoy en cambio están ansiosos por calcular complejas alteraciones del ADN de una persona.

Muchos podríamos pensar en la posibilidad de asignar este tipo de cálculos a supercomputadores como el Mare Nostrum, aunque esto es solo una solución momentánea ya que incluso los supercomputadores quedan obsoletos. No tenemos más que mirar la capacidad de calculo de un supercomputador de hace aproximadamente 10 –12 años y veremos como es similar a la capacidad de cálculo de un ordenador de hoy en día.

Por otro lado, los ordenadores son usados menos intensamente de lo que podrían serlo. De hecho, algunos artículos dicen que alrededor del 70 % del tiempo, las computadores de gama media-baja de entornos comerciales y académicos están en estado idle, mientras que estos entornos destinan un capital importante a la adquisición de estos recursos.

Si juntamos ambas ideas, podemos llegar a obtener esta nueva tecnología: los Grids computacionales. Esta tecnología consigue evitar ambas cosas, nos aporta un gran poder de cálculo y es infinitamente ampliable evitando quedar obsoleta. Por otro lado consigue aprovechar al máximo los computadores que forman parte del grid, haciéndonos por lo menos pensar que hemos amortizado bien lo pagado.

En un futuro muchas aplicaciones se adaptarán a este tipo de computación para conseguir la potencia necesaria para realizar sus cálculos. Actualmente muchos centros, por ejemplo los de investigación, no disponen individualmente de la potencia de cálculo necesaria y deben recurrir a supercomputadores para realizar sus cálculos. En un futuro se prevé que los propios centros formen grids computacionales o se unan a grids computacionales más complejos ya existentes, teniendo de esta forma siempre disponible una gran capacidad de cálculo.

Como ya se sabe una organización puede tener ocasionalmente picos de actividad, que requieren un mayor número de recursos, si las actividades pueden migrarse a máquinas que en ese momento no estuviesen en uso, los picos pasarían inadvertidos para la organización.

Pero... ¿cómo conseguir dividir un programa en diferentes ordenadores para su ejecución?. La respuesta es muy sencilla, la característica común de estos programas es que son escritos para poder particionarse en partes independientes, así un programa de uso intensivo de CPU puede pensarse como una aplicación compuesta por subtareas, cada una capaz de ejecutarse en diferentes máquinas del Grid. Sería como si para ejecutar un programa enviásemos una función a ejecutarse a cada ordenador que forma el Grid, siempre y cuando las funciones no dependieran unas de otras. De esta forma si cada función tarda n unidades de tiempo en ejecutarse y tenemos tantas máquinas en el Grid como funciones, conseguiríamos obtener el resultado en n unidades de tiempo, mientras que si solo disponemos de una máquina obtendríamos el resultado en n unidades de tiempo multiplicado por el número de funciones a computar.

1.1.2. Scheduler

Normalmente se usa la palabra aplicación para definir la pieza de trabajo de más alto nivel en un Grid. Las aplicaciones pueden descomponerse en subtareas, los cuales llamaremos tasks. El sistema Grid es el encargado de enviar cada task a un recurso para ejecutarse.

En sistema Grid más sencillo el usuario seleccionaría la máquina más adecuada para la ejecución de la tarea y luego ejecutaría un comando que enviaría el programa a la máquina seleccionada. Pero, en los sistemas Grids más avanzados esta planificación no se realiza así, para ello se dispone de planificadores más conocidos como *schedulers,* que son los encargados de decidir de forma automática la máquina más apropiada para cada tarea.

Hay diferentes tipo de schedulers, desde los más sencillos que se dedican a asignar la primera tarea que les llega a la primera máquina que encuentran disponible, hasta algunos verdaderamente complejos.

Por otro lado hay que comentar que las máquinas que forman parte de un Grid normalmente se dedican en exclusiva a él. Esto es debido a que las máquinas pueden ser obstaculizadas por el propio usuario, ya que si esa máquina estaba ocupada con un trabajo del Grid, este pasaría a un segundo plano o sería suspendido si el usuario comenzase a utilizar la potencia de cálculo de su máquina. Esta situación crea tiempos de finalización no predecibles para las aplicaciones Grid y es algo que no aporta gran fiabilidad a los schedulers. Si las máquinas son dedicadas en exclusiva al Grid se consigue que las herramientas asociadas a los planificadores computen el *completion time* muy aproximado cuando conocen las características de las máquinas que forman el Grid.

Solamente será posible una computación Grid eficiente si los schedulers son capaces de planificar lo más eficientemente posible los recursos y las aplicaciones que correrán sobre ellas.

El proceso de scheduling se realiza de forma dinámica, es decir, se realiza mientras las tareas entran al sistema, y los recursos varían su disponibilidad. Asimismo, se realiza en tiempo de ejecución, porque así se puede aprovechar de las propiedades y de la dinámica del sistema, que no son conocidas de antemano. Los planificadores dinámicos son por tanto más útiles para sistemas distribuidos reales que los planificadores estáticos.

Antes de acabar con este apartado deberíamos comentar un par de cosas más: la primera es que el escenario considerado por las heurísticas usadas en el PFC consideran que las tareas son independientes, es decir, que una tarea no depende de otra para empezar a ejecutarse. Además las tareas no pueden cambiar la máquina que les ha sido asignada una vez hayan empezado su ejecución.

En segundo lugar debemos tener en cuenta que las heurísticas que se proporcionan en la aplicación web tienen como objetivo principal minimizar el tiempo en que acaba de ejecutarse la última tarea, formalmente este objetivo se define como minimización del **makespan**. Por otro lado, como objetivo secundario se busca reducir la suma de los tiempos finales de todas las tareas, formalmente conocido como el **flowtime**.

1.1.3. Modelo de simulación basado en Matriz ETC

Para modelizar nuestro problema necesitamos una estimación o predicción de la carga computacional de cada tarea (workload) y el poder de computación de cada recurso. A partir de la capacidad de cálculo de los recursos y del workload de las tareas se puede construir una matriz ETC (Expected Time to Compute), donde cada posición $ETC_{[t][m]}$ nos indica el tiempo esperado de computación de la tarea < *t* > en el recurso < *m* >. Cada posición de la matriz puede calcularse dividiendo el workload de la tarea < *t* > entre la capacidad de computación del recurso < *m* >.

Esta formulación es factible ya que es fácil de saber la velocidad de cada recurso, y a menudo se puede saber los requisitos de computación de las tareas, ya sea a partir de especificaciones proporcionadas por el usuario, a partir de datos históricos, o de predicciones mediante herramientas especificas.

1.2. Motivación y objetivos

El propósito principal de este Proyecto Final de Carrera es diseñar e implementar una aplicación web que ofrezca unos servicios computacionales en Internet, con el fin de resolver problemas de Scheduling. Uno de los requisitos del proyecto es que la aplicación web tiene que ser accesible desde cualquier plataforma y con cualquier navegador.

En la actualidad se dispone de un seguido de heurísticas para solucionar problemas de Scheduling para entornos Grids, realizadas en un proyecto de fin de carrera anterior[1]. Así, como se ha comentado anteriormente este proyecto consiste en el desarrollo de una aplicación web que permita la ejecución remota de las heurísticas encargadas de solucionar los problemas de Scheduling.

La existencia de esta aplicación viene motivada por varias razones:

- Permite la ejecución de las heurísticas a través de una interfaz amigable y sencilla. Es más fácil para el usuario ejecutarlas en un entorno gráfico que desde la línea de comandos.

- Permite la ejecución de las heurísticas remotamente desde cualquier máquina que disponga de un navegador y acceso a Internet. Así se consigue que la posibilidad de solucionar los problemas de Scheduling llegue a un mayor números de personas, ya que no es necesario que tengan instalado ningún programa adicional en su máquina.

- Por último facilita la extracción de resultados permitiendo exportarlos para un posterior tratamiento estadístico.

[1] Proyecto fin de carrera realizado por Javier Carretero.

1.3. Alcance del proyecto

En este punto se explica a quién va dirigido el proyecto.

La aplicación web se encuentra disponible para todo el mundo que quiera acceder a ella. Aunque está especialmente destinada a aquellos que tengan interés en la ejecución de las heurísticas destinadas a la resolución de problemas de Scheduling, o más en general, a todas las personas que sientan un especial interés por esta nueva tecnología: Los Grids Computacionales.

La aplicación también va destinada para estudiantes de ingenierías que necesitan estudiar el comportamiento de los Grids, ya sea para prácticas de alguna asignatura o por cualquier otro motivo.

Por último la aplicación puede ser útil para algunos investigadores, que necesiten solucionar sus problemas de Scheduling, ya sea para estudios estadísticos o para simulaciones de los entornos Grids.

1.4. Proyectos relacionados

- AELOUS: La reciente explosión de crecimiento de Internet da salida a la posibilidad de un computador global de gran escala formado por entidades computacionales conectadas a Internet(possible movimiento, con variación de las capacidades computacionales, conectadas entre si con diferentes medios de comunicación), globalmente disponible y capaz de proveer a sus usuarios un enriquecido menú de servicios de alto nivel integrados para hacer uso de su agregado poder computacional, espacio de almacenamiento y recursos de información. Realizar esto eficientemente y transparentemente es un gran desafío que puede ser logrado introduciendo una capa intermedia, la computadora overlay.

El objetivo de AEOLUS es investigar los principios y desarrollar métodos algorítmicos para construir algo parecido a una computadora overlay que permita este eficiente y transparente acceso a los recursos de una computadora global basada en Internet.

- ASCE (Autoorganización en Sistemas de Comunicación Emergentes):

Número: TIN2005-09198-C02-02

Descripción:

Periodo: 1/2006-1/2009

Capital: 63.070 euros

Miembros: M. Serna (local coordinator), C. Àlvarez, M.J. Blesa, J. Gabarró, F. Xhafa.

Participantes:
Grupo de Sistemas y Comunicaciones (GSyC) (Universidad Rey Juan Carlos), Departament de Llenguatges i Sistemes Informàtics (Universitat Politècnica de Catalunya).

Coordinador: A. Fernández (Madrid)

2. Heurísticas y Metaheurísticas tratadas en el PFC

La planificación eficiente de trabajos independientes en un sistema de computación distribuido heterogéneo, como es el Grid, es de vital importancia si se quiere obtener un buen uso de los recursos que forman el sistema. Sin embargo, encontrar una planificación óptima en dicho sistema se ha demostrado que es por lo general, NP – difícil.

A lo largo de este apartado se describirán un seguido de heurísticas especificas (ad-hoc) de las cuales su característica más importante es su bajo coste computacional, por lo que pueden ser usadas para encontrar soluciones iniciales para metaheurísticas más complejas o bien como soluciones en caso que no sea factible utilizar las metaheurísticas.

Una vez explicadas las heurísticas especificas se describirán de forma breve las dos metaheurísticas tratadas también el proyecto: metaheurísticas de Algoritmos Genéticos y metaheurísticas de Búsqueda Tabú.

2.1. Heurísticas Ad-hoc

Las heurísticas específicas que se van a describir a continuación, pueden ser agrupadas en dos modos de ejecución: heurísticas de modalidad inmediata (**immediate mode**) y heurísticas de modalidad *batch* (**batch mode**), o por lotes.

En el modo inmediato, una tarea es planificada en un recurso tan pronto como entra en el planificador. En la modalidad *batch*, las tareas no se planifican en cuanto llegan al sistema, sino que son agrupadas en un *pool* el cual es planificado cada cierto intervalo de tiempo especificado por un *mapping event*.

Por tanto, la estrategia *immediate* es en principio beneficiosa cuando no se desea que las tareas tengan que esperar al siguiente evento de la planificación. En la modalidad batch, el planificador considera un conjunto de tareas para ser planificadas. Esto proporciona una gran ventaja, ya que da mayores posibilidades a las heurísticas de planificación de hacer mejores decisiones que las heurísticas de modo immediate.

A continuación se describen diez heurísticas ad-hoc de las cuales cinco pertenecen al modo immediate y las cinco restantes pertenecen al modo batch.

2.1.1. Heurísticas Ad-hoc de modo inmediato

En este apartado se describen las heurísticas de modo inmediato. Estas son: *Minimum Completion Time, Minimum Execution Time, Switching Algorithm, K-Percent Best* y *Opportunistic Load Balancing*.

- ***OLB:*** La heurística *OLB (Opportunistic Load Balancing)* asigna una tarea a la máquina que queda desocupada más tempranamente, sin considerar el tiempo de ejecución de la tarea en esa máquina. Si varios recursos quedan libres al mismo tiempo, entonces se escoge uno de ellos arbitrariamente. Esta heurística suele implementarse de la siguiente manera: las máquinas desocupadas acceden a una cola global de tareas que están esperando ser ejecutadas, y escogen una tarea. El tiempo necesario para asignar una tarea depende también de la implementación.

- ***MCT:*** La heurística *MCT* (M*inimum Completion Time)* asigna cada tarea a la máquina que resulta ser la que le proporciona el tiempo de finalización más temprano (*earliest completion time)*. Esto causa que algunas tareas se asignen a máquinas que no tienen el mejor tiempo de ejecución para ellas.

- ***MET:*** La heurística MET (M*inimum Execution Time)* asigna cada tarea a la máquina que ejecuta la tarea en la mínima cantidad de tiempo posible.

  Esta heurística puede provocar un desbalanceo de carga muy acentuado, en caso de que los recursos presentes en el *Grid* sean muy dispares en su poder de computación. La ventaja de este método es que cada tarea va a parar a la máquina a la cual tiene más afinidad.

- ***SA:*** La heurística S*witching Algorithm* surge al observar los siguientes hechos: la heurística *MET* potencialmente crea un desbalanceo de carga importante al asignar muchas más tareas a unas máquinas que a otras, mientras que *MCT* intenta balancear la carga debido a que se guía de los tiempos de finalización de las máquinas (*completion times).*

  La idea de esta heurística es usar *MET* a expensas del balanceo de carga, hasta que se llegue a un cierto *threshold*, a partir del cual se usará *MCT* para nivelar la carga entre las máquinas.

- ***KPB:*** La heurística *KPB (K-Percent Best)* considera un subconjunto de recursos del conjunto global para mapear una tarea. El propósito de la heurística es asignar una tarea en una buena máquina candidata, y además, evitar asignarla en una máquina que sea más adecuada para alguna de las tareas que aún están por asignar.

Ejemplo de funcionamiento de las heurísticas de tipo "Immediate mode"

A modo de ejemplo del funcionamiento de las heurísticas descritas hasta el momento, imaginemos un sistema con tres recursos, m_0, m_1 y m_2 con una carga actual (*ready_time*) de:

m_0	m_1	m_2
75	*110*	*200*

A continuación veremos un ejemplo de rendimiento para un caso muy sencillo en el que se planifican tres tareas t_0, t_1 y t_2, las cuales llegan en este orden. La siguiente tabla muestra el tiempo esperado de ejecución (*expected time to compute*) de cada una de las tareas en cada uno de los recursos existentes en el sistema.

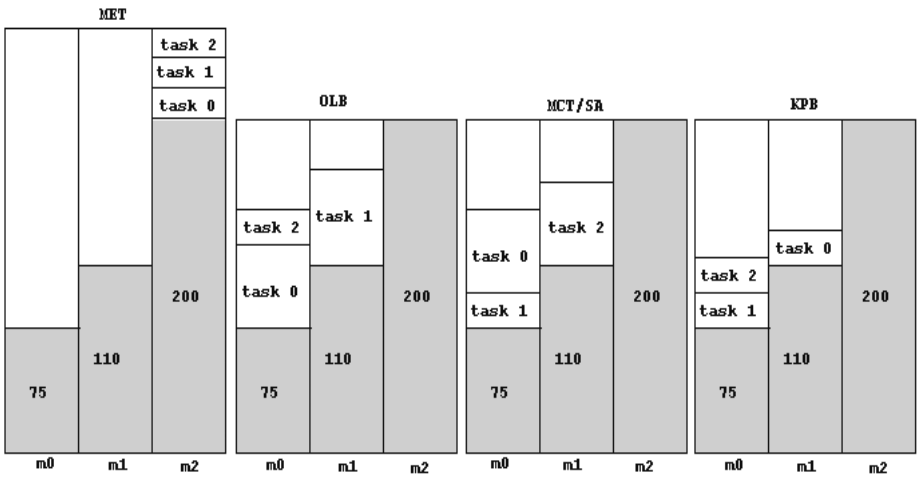
	m_0	m_1	m_2
t_0	*50*	*20*	*15*
t_1	*20*	*60*	*15*
t_2	*20*	*50*	*15*

La heurística *OLB* asigna t_0 *a* m_0, ya que m_0 es el recurso que antes estará libre *(idle*). Similarmente, asigna t_1 y t_2 a m_1 y m_0 respectivamente. El tiempo en el que las tres tareas habrán acabado de ejecutarse es de *170* unidades de tiempo.

La heurística *MCT* determina que el tiempo mínimo de finalización para t_0 se consigue en la máquina m_0, y realiza esta asignación, a pesar de que el tiempo de ejecución de t_0 sea más del triple que en m_2. A continuación *MCT* asigna t_1 en m_0 y t_2 en m_1, proporcionando un *makespan* de *160* unidades.

La heurística *MET* descubre que todas las tareas tienen su menor tiempo de ejecución en m_2, sin tener en cuenta que m_2 ya está seriamente cargada, y asigna las tres tareas en dicha máquina. Como consecuencia, el tiempo total de finalización es de *245* unidades.

El procedimiento seguido por las heurísticas SA y KPB no lo describiré debido a que se salen del alcance del proyecto, pero de todas maneras mostraré sus resultados.



2.1.2. Heurísticas Ad-hoc de modo batch

En este apartado se describen cinco heurísticas de modo *batch*. Estas son: *Min-min, Max-min, Sufferage, Relative Cost y LJFR-SJFR.*

- ***Min-min:*** La heurística *Min-min* en la fase de inicialización calcula los valores $completion_{ij}$ para cada tarea t_i y cada máquina m_j usando los valores etc_{ij} y $ready_j$. Para cada tarea t_i, la máquina que le ofrece el tiempo más temprano de finalización es determinado recorriendo la fila *i* de la matriz *completion.* Una vez que se ha calculado para cada tarea la máquina que la finalizará de ejecutar más pronto, se elige la tarea que tenga el tiempo de finalización más temprano entre todas las tareas; para mapearla a su correspondiente máquina.

  Finalmente, se añade su *etc* a los valores de $completion_{ij}$ para que todas las tareas t_i que quedan por planificarse la tengan en cuenta.

- ***Max-min:*** La heurística *Max-min* es muy similar a la heurística *Min-min.* Se diferencia de *Min-min* en que una vez que se ha calculado para cada tarea que queda por planificar la máquina que la acaba de ejecutar más rápidamente, se planifica la tarea t_k que tiene el tiempo de finalización máximo, y se mapea a la máquina correspondiente.

  Max-min se comporta mejor que *Min-min* en casos en los que hay muchas más tareas pequeñas que tareas grandes.

- ***Sufferage:*** La heurística *Sufferage* se basa en la idea de que se pueden generar mejores planificaciones asignando a una máquina la tarea que sufriría más en términos de *completion time* (tiempo de finalización) si no se le asignará a dicha máquina.

  Se define el valor de ***sufrimiento*** de una tarea t_i la diferencia entre el segundo tiempo de finalización más temprano (en una máquina m_y) y el primero tiempo de finalización más temprano (en una máquina m_x).

  En caso de que m_x esté ocupada por una tarea y el planificador decidiera enviar otra tarea a la máquina, las dos tareas competirán por ella. La tarea que tenga un mayor índice de "*sufrimiento"* adquirirá la máquina.

- ***Relative-Cost:*** La heurística *Relative Cost* (*RC*) intenta encontrar un compromiso entre el balanceo de carga y entre el grado de proximidad en la asignación de los trabajos a sus recursos preferidos (en el sentido de tiempo mínimo de ejecución). Para ello *Relative Cost* no usa simplemente el criterio del *completion time* para asignar una tarea, sino que también tiene en cuenta otros factores.

  El buen rendimiento de los algoritmos de planificación es determinado, en parte, por la proporción de tareas que son capaces de asignar a las máquinas que las ejecutan más rápido.

- ***LJFR-SJFR:*** La heurística *LJFR-SJFR* (*Longest Job to Fastest Resource – Shortest Job to Fastest Resource)* difiere de las heurísticas que se han explicado hasta el momento.

  El objetivo de esta heurística es el de minimizar el *makespan* y el *flowtime* simultáneamente. Para ello, se alterna la heurística *LJFR* (con el objetivo de reducir el *makespan*) con la heurística *SJFR* (para reducir el *flowtime*).

  La heurística comienza ordenando los trabajos en orden creciente de *workload*. Inicialmente se disponen de *m* máquinas libres, y son sobre estas máquinas sobre las que se alocatan las *m* tareas más "pesadas" (la tarea más grande va a la primera máquina más rápida, la segunda tarea más grande va a la segunda máquina más rápida, etc…).

  Cada vez que uno o más trabajos acaben su ejecución, se alternará el uso de la heurística *LJFR* con la heurística *SJFR*. Esto es equivalente a decir que a cada paso se elegirá la máquina más rápida que acaba de ejecutar sus tareas antes y se le asignará o bien un trabajo corto (*SJFR*), o bien un trabajo grande (*LJFR*).

Ejemplo de funcionamiento de las heurísticas de tipo "Batch mode"

A modo de ejemplo del funcionamiento de las heurísticas descritas hasta el momento, imaginemos un sistema con cuatro recursos, m_0, m_1, m_2 y m_3 con una carga inicial (*ready_time*) de cero unidades de tiempo en todas las máquinas.

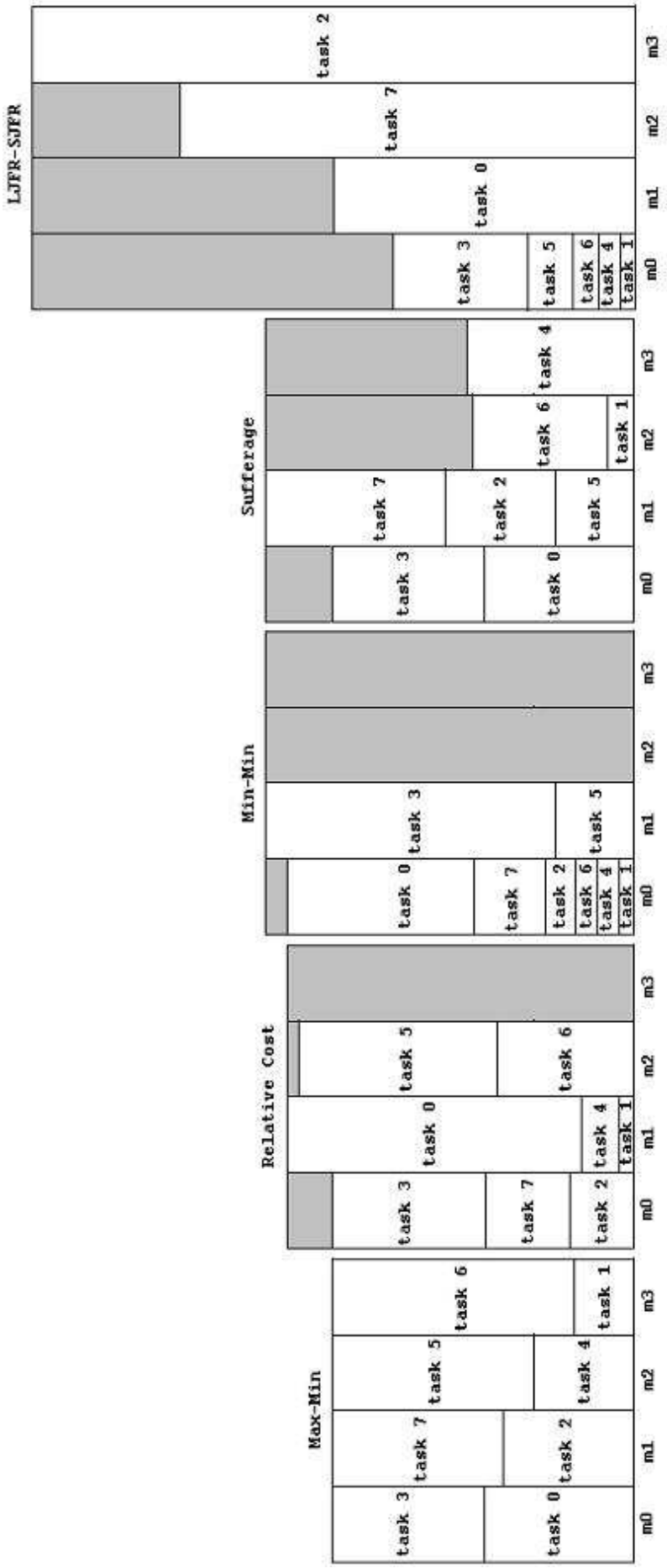
A continuación veremos un ejemplo de rendimiento en el que se planifican ocho tareas. Las siguientes tablas muestran la estimación de carga de trabajo de cada tarea, la estimación de la capacidad de cómputo de cada recurso y la matriz *etc* calculada a partir de las dos tablas:

t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
500	*20*	*200*	*500*	*60*	*140*	*80*	*300*

m_0	m_1	m_2	*m3*
10	*5*	*2*	*1*

	m_0	m_1	m_2	m_3
t_0	*50*	*100*	*250*	*500*
t_1	*2*	*4*	*10*	*20*
t_2	*20*	*40*	*100*	*200*
t_3	*50*	*100*	*250*	*500*
t_4	*6*	*12*	*30*	*60*
t_5	*14*	*28*	*70*	*140*
t_6	*8*	*16*	*40*	*80*
t_7	*30*	*60*	*150*	*300*

Las planificaciones generadas por cada una de las heurísticas de modo *batch* se resumen en la siguiente figura. El ***makespan*** es respectivamente de *100, 116, 128, 128* y *200*.


LJFR-SJFR
Sufferage
Min-Min
Relative Cost
Max-Min
m0
m1
m2
m3

2.2. Metaheurísticas: Algoritmos Genéticos

Un algoritmo genético (GA) es una técnica de búsqueda usada en computación para encontrar soluciones optimas o muy próximas a la optima. Los algoritmos genéticos se catalogan como heurísticas de búsqueda.

Hay que destacar que los algoritmos genéticos son una clase particular de los algoritmos evolutivos.

Los algoritmos genéticos son implementados como una simulación de computador, en la cual hay una población de representantes o posibles soluciones (llamadas los cromosomas o los genotipos) que se van evaluando una por una. Tradicionalmente las soluciones se representan en binario como cadenas de 0s y de 1s, pero otras codificaciones son también posibles. El proceso empieza con una población de individuos aleatoriamente generados que son tratados como posibles soluciones. En cada iteración la aptitud de cada individuo frente al problema se evalúa.

Comúnmente, el algoritmo termina cuando un número máximo de generaciones se ha producido, o un nivel satisfactorio de la solución ha sido alcanzado. Si el algoritmo ha terminado debido a un número máximo de generaciones es muy probable que no se haya encontrado la solución más optima posible del problema.

Un algoritmo genético típico requiere de dos cosas básicas para ser definido:

1. Una representación para las posibles soluciones.
2. Una función encargada de evaluar la calidad de la solución encontrada.

Una representación estándar de la solución es por ejemplo un vector de bits. La función se define sobre la representación genética y mide la calidad de la solución representada. La función es siempre dependiente del problema y por norma general una función de evaluación no puede ser utilizada para dos problemas heurísticos diferentes.

Por ejemplo en el problema más que conocido de la mochila el objetivo es maximizar el número de objetos que podemos transportar o poner en nuestra mochila de una cierta capacidad prefijada. Una representación de una solución puede ser un vector de bits, donde cada bit representa un objeto diferente, y el valor del bit representado nos indica si el objeto se encuentra en la mochila o no. En este problema no todas las representaciones de la solución son válidas, ya que hay soluciones que pueden exceder el tamaño de la mochila. Para valorar cual es la mejor opción necesitaremos valorarlo mediante la función evaluadora, la cual si encuentra una solución no válida retornará un cero y si encuentra una solución válida retornará un valor indicado la calidad de la solución.

Una vez que se haya realizado la representación genética y se tenga implementada la función de evaluación se deben empezar a producir las soluciones aleatoriamente.

El número de soluciones y el tamaño de las mismas depende de la naturaleza del problema, pero típicamente suelen crearse centenares o millares de soluciones posibles. Tradicionalmente estas soluciones son, como hemos comentado, creadas aleatoriamente pero también se da el caso en el que las soluciones son *sembradas* en áreas donde se encuentran posiblemente las soluciones más optimas.

Hay algoritmos encargados de buscar las llamadas soluciones sembradas, pero sin embargo hay otros que buscan soluciones cogiendo muestras al azar, este proceso puede ser muy desperdiciador de tiempo.

Una vez se ha realizado un primer barrido de las soluciones, se han escogido las que mejor resultado han dado en la función evaluatoria. Éstas ahora serán tratadas como padres y se intentará sacar de ellas otras posibles soluciones que mantengan los beneficios de la progenitora pero con pequeñas mutaciones que puedan hacer mejorar su resultado. Este proceso es repetido una y otra vez, hasta que se llega a una solución optima o el algoritmo se acaba.

Hay que destacar que el método de *reproducción* explicado arriba suele aumentar considerablemente la media de los resultados obtenidos en la función de evaluación ya que las peores soluciones han sido eliminadas de la muestra a tratar.

Se repite este proceso generacional hasta que se ha alcanzado una de las condiciones de salida del algoritmo genético. Los condiciones más habituales para salir de este tipo de algoritmo son las mencionadas a continuación:

- Se encuentra una solución que satisface los criterios mínimos establecidos.
- Se han alcanzado el número fijo de generaciones prefijadas antes de la ejecución.
- Puede darse el caso de que se limite el tiempo de computo (por motivos económicos, por ejemplo) a un tiempo limitado.
- Se ha alcanzado de más alta graduación posible o las iteraciones sucesivas no producen mejores resultados.
- Se detiene manualmente el algoritmo.

2.2.1. Parámetros de configuración

Los parámetros que podemos configurar en los algoritmos de Tabú Search son los siguientes:

Number of tasks: Por ejemplo 128, el máximo son 512.

Number of machines: Por ejemplo: 12, el número máximo de máquinas son 16.

Independents runs: Por ejemplo: 1.

Max_time: Por ejemplo 90, , este parámetro está limitado por el administrador de la página y su máximo puede variar según lo decida el administrador.

Evolution_steps: Por ejemplo 12000.

Pop_size: Por ejemplo 10.

Intermediate_size: Por ejemplo 10.

Prob_cross: Por ejemplo 0.9.

Prob_mutate: Por ejemplo 0.4.

Start_choice: Por ejemplo :StartLJFRSJFR.

Select_choice: Por ejemplo SelectLinearRanking.

Select_extra: Por ejemplo 0.70.

Cross_choice: Por ejemplo CrossOnePoint.

Cross_extra: Por ejemplo 0.50.

Mutate_choice: Por ejemplo MutateMove.

Mutate_extra: Por ejemplo 0.75.

Replace_if_better: Por ejemplo true.

Replace_generational: Por ejemplo false.

Struggle_replace: Por ejemplo false.

Struggle_extra: Por ejemplo 2.

2.3. Metaheurísticas: Búsqueda Tabú

La búsqueda tabú es una metaheurística que guía el procedimiento de la búsqueda local heurística a explorar el espacio de soluciones además de la optimización local.

Se diferencia del algoritmo genético (GA) porque la cerca tabú incluye un mecanismo de memoria. Según la idea de Glover, quien propuso el método en los años 80, para solucionar un problema usando la búsqueda tabú se han de definir los siguientes componentes:

- Configuración: Es una solución o asignación de valores a unas variables.

- Movimiento: Un movimiento caracteriza el proceso de generar una solución factible al problema a partir de la solución actual del mismo, el movimiento es un proceso por el cual se genera una nueva solución partiendo de la actual.

- Vecindad: Un vecino de la solución es un conjunto de todos los posibles abandonamientos de la configuración actual. Hay que notar que esta configuración depende de la implementación y la naturaleza del problema.

- Condiciones Tabú: Por tal de evitar una búsqueda ciega, la técnica de la búsqueda tabú usa un conjunto de restricciones definidas por el problema específico, conocidas como las condiciones tabú. Son unas ciertas condiciones impuestas a los movimientos que hacen que se produzcan nuevas soluciones. Estos movimientos prohibidos son conocidos como tabú. Se hacen formando una lista de cierto tamaño donde se registran los movimientos que están prohibidos.

- Condiciones de aspiración: son reglas que predominan sobre las restricciones tabú, es decir si un movimiento está prohibido por una restricción tabú, cuando se satisface un criterio de aspiración puede hacer que este movimiento vuelva a ser permitido.

Con estos componentes descritos, el algoritmo de búsqueda tabú puede ser descrito como:

(i) Comenzar con una cierta configuración y evaluar la función de criterio para esta configuración.

(ii) Seguir un vecino de la configuración actual, es decir, un conjunto de movimientos candidatos. Si el mejor de estos criterios no es tabú o si es tabú pero satisface el criterio de aspiración, seleccionamos este movimiento y lo consideramos como la nueva configuración de la solución, en caso contrario seleccionamos el mejor movimiento que no sea tabú y lo consideramos a éste como la nueva configuración de la solución.

(iii) Repetir los pasos (i) y (ii) hasta que se cumpla el criterio de terminación.

La mejor solución en la última vuelta es la solución obtenida por el algoritmo. Hay que remarcar que los movimientos seleccionados en cada iteración son añadidos a la lista de tabú con el fin de que no sean permitidos por ser contradictorios en las siguientes iteraciones.

Hay que mencionar también que la lista tabú tiene un cierto tamaño, y que cuando la longitud de la búsqueda tabú llega al tamaño máximo establecido y se produce un nuevo movimiento este se añade a la lista tabú, obligando en ese momento a eliminar de la lista el primer movimiento que fue introducido. Así podríamos decir que estamos tratando con una lista circular.

La lista tabú tal como se ha comentado anteriormente tiene un seguido de ventajas:

1) La búsqueda tabú evita las trampas i continua la búsqueda para dar una solución final que se acerca más a la más optima.

2) La búsqueda tabú es muy general y conceptualmente es mucho más simple en que algoritmo genético (GA)

3) La búsqueda tabú se establece en un marco flexible de una variedad de estrategias originadas en la inteligencia artificial y por lo tanto esta abierta a mejoras adicionales.

2.3.1. Parámetros de configuración

Los parámetros que podemos configurar en los algoritmos de Tabú Search son los siguientes:

Number of tasks: Por ejemplo 128, el máximo son 512.

Number of machines: Por ejemplo: 12, el número máximo de máquinas son 16.

Independents runs: Por ejemplo: 1

Number iterations: Por ejemplo 110, este parámetro está limitado por el administrador de la página y su máximo puede variar según lo decida el administrador.

Tabu Size: Por ejemplo 900

Maximum tabu Status: Por ejemplo: 32

Aspiration value: Por ejemplo: 33

Maximum repetitions: .Por ejemplo: 64

Number of intensifications: Por ejemplo: 5

Number of diversifications: Por ejemplo: 5

Elite Size: Por ejemplo: 10

Max nb swaps: Por ejemplo: 100

Max nb Transfer: Por ejemplo: 300

Max Load Interval: Por ejemplo: 1

Percent min Load: Por ejemplo: 1

Max Time to spend: Por ejemplo: 90, este parámetro está limitado por el administrador de la página y su máximo puede variar según lo decida el administrador.

3. Especificación

Los servicios computacionales que se ofrecen desde la aplicación web son heurísticas que tienen como función planificar tareas en entornos distribuidos como son los *Grids Computacionals*. Estas heurísticas serán tratadas como cajas negras, es decir, no entraremos en el análisis de sus funcionalidades ni tampoco en como están implementadas, así solo haremos un estudio de los parámetros de entrada y salida que necesitan/retornan las ejecuciones que realizaremos sobre ellas. La implementación de éstas fueron realizadas en C++ y sobre Linux.

Una vez que los usuarios acceden a la aplicación web, éstos dispondrán de las implementaciones de las heurísticas y de un conjunto de máquinas para obtener la solución a sus problemas de scheduling. La interfaz web además facilitará información al usuario sobre los problemas que se pueden tratar en la aplicación y mostrará una lista de las heurísticas que están implementadas en el departamento del LSI para resolver este tipo de problemas.

El usuario tendrá que introducir los datos necesarios para que la ejecución de los problemas se pueda llevar a cabo y que éstos sean solucionados por las heurísticas. Los datos que debe introducir el usuario se dividen en dos partes, la primera es la instancia propia del problema a resolver, consiste en una matriz que representa los valores de la *matriz ETC*. La segunda parte de los datos es introducir todos aquellos parámetros que puedan influir en la ejecución de la heurística seleccionada como puede ser: el número de máquinas, el tipo de heurística, etc...

Para facilitar esta labor al usuario, la interfaz web proporcionará ejemplos de los posibles valores que pueden asignarse a los parámetros, así como del contenido que tendrá que tener la instancia proporcionada para la ejecución.

Por otro lado hay que destacar que el espacio web deberá ir dividido en dos partes bien diferenciadas, en primer lugar tendremos la parte pública a la cual tendrán acceso todos aquellos que accedan a la página web. Después encontraremos la parte privada de la aplicación, en la cual el uso está restringido solo a aquellos usuarios que estén registrados.

Espacio público: Es la encargada de dar la bienvenida a todos los usuarios de la aplicación, en ella encontraremos documentación e información sobre las heurísticas tratadas en el proyecto. El usuario también podrá encontrar ejemplos de instancias, de resultados obtenidos después de la ejecución de las heurísticas y de información acerca de los problemas de Scheduling. Uno de los apartados más importante de la zona pública de la web es que el usuario podrá ejecutar y familiarizarse con una pequeña demo de la aplicación que encontrará en la parte privada de la web.

Por otro lado encontraremos un apartado de alta para nuevos usuarios en la cual el usuario tendrá como campos obligatorios tres campos: usernarme, password y e-mail. Existirá otro apartado dedicado a la recuperación de contraseña para usuarios registrados con anterioridad donde mediante el username se podrá obtener el password, siendo éste enviado a la cuenta de correo que el usuario indico cuando realizó su registro.

Espacio privado: A esta zona de la aplicación solo podrán acceder los usuarios registrados. El acceso a este espacio se realizará desde la página de bienvenida, y se solicitará la identificación del usuario mediante un username y un password.

Esta zona de la aplicación tendrá tres formatos diferentes, dependiendo del tipo de usuario que acceda a ella. Así diferenciaremos entre tres tipos de usuarios:

- **Invitado**: Como es lógico, será el que menos privilegios tendrá. Podrá ejecutar un número limitado de instancias y en un tiempo limitado. Podrá configurar todos los datos que sean configurables en el programa de Scheduling y obtendrá los resultados de la forma que lo deseé.

- **Investigador**: Dispondrá de las utilidades y privilegios que tienen los invitados, pero además tendrá permitido ejecutar más instancias en el programa y durante más tiempo. El investigador tendrá la posibilidad de proponer instancias creadas por el propio usuario para que se hagan públicas en la web.

- **Administrador**: Podrá gestionar información del resto de usuarios (invitados e investigadores), decidirá los tiempos de ejecución máximos para las instancias de cada tipo de usuario además del número máximo de instancias a ejecutar en un intervalo de tiempo. El administrador dispondrá de un apartado de estadísticas donde podrá observar el uso que se está realizando de la aplicación. En las mismas estadísticas podrá consultar datos como el número de accesos a la página en un día, semana o mes. Así como el número de usuarios registrados en la aplicación.

  Para finalizar el administrador podrá ver las propuestas realizadas por los investigadores y si lo desea podrá seleccionar la instancia para mostrarla como *instancia pública.*

Por otro lado deberíamos señalar que todos los usuarios registrados dispondrán de un apartado donde podrán realizar tareas de gestión de sus cuentas. En este apartado podrán modificar su contraseña, cambiar su dirección de e-mail o darse de baja en la aplicación.

La zona ejecutable deberá mostrar un pequeño historial con las ultimas ejecuciones que el usuario ha realizado en el programa, el número de ejecuciones que podrá observar el usuario estarán limitadas por el administrador, el cual podrá configurarlas en un apartado sobre la configuración de los usuarios. Además los usuarios dispondrán de un apartado donde los usuarios podrán encontrar ejemplos de ejecuciones, con el fin de familiarizarse con la aplicación.

Los usuarios podrán introducir los valores de los parámetros de las heurísticas mediante un formulario que será ofrecido por la aplicación web. El usuario visualizará en el formulario los campos necesarios para la ejecución de la heurística, así como una pequeña explicación del parámetro correspondiente.

El usuario también debe decidir cual será la instancia del problema que deberán tratar las heurísticas. Esta decisión se puede hacer de dos formas, la primera es seleccionar la instancia de un desplegable el cual será mostrado por la aplicación, este desplegable contiene las denominadas *instancias públicas*. Por otro lado, como segunda opción el usuario puede

crear su propia instancia , respetando el formato de instancia que se explica en la parte pública de la aplicación, en un fichero y después seleccionándolo cuando la aplicación web lo solicite.

A lo largo del formulario de configuración de parámetros al usuario se le preguntará por la manera en que desea recibir los resultados de la ejecución. Éstos podrán ser mostrados de dos maneras: Enviando los resultados a la dirección de correo electrónico que el usuario haya introducido en el registro o mostrando los resultados por la pantalla.

La primera de las opciones permite recibir unos resultados más completos, ya que la muestra por pantalla de resultados se ha limitado debido a la gran cantidad de información que se podía llegar a mostrar, así solo se mostraran los parámetros de medida de las ejecuciones: Flowtime, Makespan, etc...

3.1. Software utilizado para el desarrollo de la aplicación Web

A continuación se nombran y describen algunos de los programas y sistemas operativos que han sido utilizados durante la realización de este proyecto, ya sea en el apartado de implementación, diseño o escritura de la memoria.

En primer lugar nombraré los programas sobre los cual va a ejecutarse la aplicación que se ha diseñado en el PFC, son los siguientes: Apache 2.x, PHP5 y MySQL. Todos son de libre distribución, el motivo por el cual hemos seleccionado estos programas son los siguientes:

Apache fue seleccionado por ser un servidor fiable, usado en otros proyectos obteniendo buenos resultados y por ser de bajo coste ya que es de libre distribución. La versión de Apache que utilizaremos es la 2.0.59 ya que es una de las versiones más modernas y fiables que podemos descargar de su página web: *http://httpd.apache.org*

PHP5 lo seleccionamos porque nos permite que la aplicación web se pueda ejecutar independientemente de la plataforma desde donde sea solicitada la información, ya que ésta se realiza mediante un script el cual se ejecutará en el servidor, así el usuario únicamente obtendrá y verá en su computador los resultados de dicha ejecución. La versión utilizada es PHP 5.2.1, el motivo por el cual se ha seleccionado ésta es porque era la última versión disponible en la página oficial en el momento de iniciar el proyecto: http://www.php.net

La elección de la base de datos **MySQL**, se debe a que era necesario una base de datos relacional para almacenar algunos datos de las ejecuciones y de los registros de usuarios. MySQL se adaptaba a nuestras necesidades y además en su página podemos encontrar versiones antiguas para descargar gratuitamente: http://www.mysql.com

Por otro lado tenemos que destacar que las ejecuciones se realizarán de forma asíncrona en la aplicación web. La ejecución se realizará en procesos batch que estarán controlados por un planificador de tareas. El planificador de tareas que se utilizará es el **CONDOR** de la Universidad de Wisconsin (http://www.cs.wisc.edu/condor), propuesto por el departamento del Lenguajes y Sistemas Informáticos y instalado en el LCLSI.

El software Condor fue creado para optimizar la ejecución de procesos batch en máquinas que forman parte de un cluster y de esta manera aprovechar la potencia que ofrecen. El uso que daremos del software Condor en nuestra aplicación es únicamente para el control de ejecuciones. Cada vez que un usuario ejecute nuestra aplicación, ésta hará una llamada al software Condor con el proceso, el software Condor detectará la ejecución y mirará si la máquina donde ha de ejecutarse el proceso está disponible o no, en caso de que no lo este ofrecerá la posibilidad de continuar la ejecución del proceso en otra máquina del Grid computacional, evitando la saturación de las maquinas donde se encuentra instalado el programa de Scheduling.

Por otro lado hay que mencionar programas de menos relevancia, pero que sin ellos no hubiese sido posible la realización del proyecto:

Sistemas operativos:

- **Windows XP**: utilizado en parte del desarrollo del proyecto y en las pruebas. También ha sido utilizado para redactar la memoria.

- **Linux Ubuntu 5.1**: utilizado en parte del desarrollo del proyecto y donde se han realizado todas las compilaciones de los programas ofrecidos en la aplicación web.

Programas de desarrollo:

- **PHP Edit 5.0**: Programa de desarrollo de aplicaciones PHP, utilizado en la parte inicial de la implementación, se dejo de lado el programa debido a que era una versión de prueba únicamente para 30 días.

- **NotePad 2**: Substituto del programa anterior, en el se ha desarrollado la mayoría de las páginas PHP que forman la aplicación web. Podemos encontrarlo de forma gratuita en la siguiente dirección: http://www.flos-freeware.ch/

- **Microsoft Word**: Se ha utilizado para redactar la memoria, además de Word se han utilizado otros programas incluidos en el paquete de ofimática de Microsoft, como por ejemplo Excel para la realización de tablas.

Programas de ayuda a la conexión:

- **Putty SSH**: Es un programa encargado de realizar una conexión remota y segura via SSH con un servidor. Ha sido utilizado para subir los datos necesarios al servidor donde quedará alojado el proyecto. Este programa solo funciona en modo consola.

- **WinSCP 3.8.2**: Es un programa encargado de realizar conexiones remotas, en este caso hablamos de un programa con una interfaz gráfica muy agradable el cual ha sido utilizado también para subir ficheros a servidores remotos. Podemos encontrarlo de forma gratuita en la siguiente dirección: http://winscp.net

Navegadores de acceso a Internet:

- **Internet Explorer 6.0 / 7.0**: Utilizado para realizar las pruebas de la aplicación, además de para conseguir diversa información para la realización de la memoria.

- **Mozilla Firefox 2.0**: Al igual que el Internet Explorer, este navegador ha sido utilizado para realizar parte de las pruebas del proyecto, verificando que el proyecto responde de la misma manera ante diferentes navegadores.

Para finalizar el apartado de programas utilizados comentaremos que para ejecutar la aplicación web a nivel de usuario no necesitaremos ningún programa de los mencionados anteriormente. Simplemente tendremos que disponer de un navegador de páginas webs como pueden ser el Internet Explorer o el Mozilla Firefox. La aplicación garantiza el acceso desde cualquier S.O. y los requisitos a nivel de hardware son mínimos, ya que la aplicación funciona en cualquier ordenador que sea capaz de acceder a Internet sin problemas.

Por otro lado destacar que la página se ha pensado de forma que la carga de su contenido no sea una operación pesada, de esta forma se garantiza que incluso con las conexiones más lentas a Internet, la página realizará una carga relativamente rápida.

4. Diseño y estructuración

En este apartado se explicará la estructura que tendrá la interfície web. Esta estructura se comentará desde diferentes puntos de vista como:

1. La arquitectura interna que tendrá la web, es decir, como estarán organizados sus directorios y el contenido que podemos encontrar dentro de cada uno de ellos.

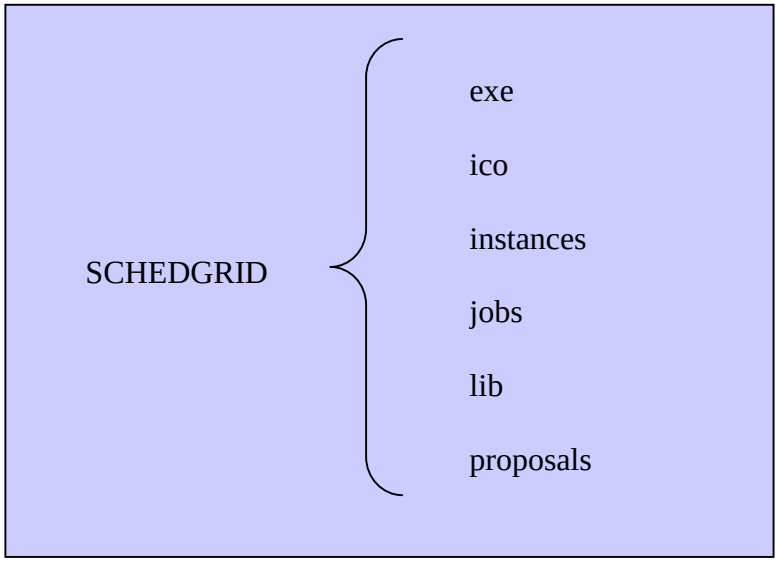
2. Los flujos de trabajo, es decir, el camino que deben seguir los diferentes tipos de usuarios para poder ejecutar y navegar por la página web.

3. El diseño gráfico que tendrá la interfície web.

4.1. Estructura de directorios

Se hace necesario crear una estructura de directorios con el fin de mantener una cierta organización interna de los ficheros que hace servir la interfície web. Además esta estructura también facilita al administrador de la página el acceso a la información que se ha ido recopilando a lo largo de los accesos de los usuarios o para realizar posibles cambios en la página web.

La estructura de directorios queda de la siguiente manera:



El directorio **SCHEDGRID** contendrá los ficheros públicos, a los cuales el usuario podrá acceder a través de su navegador habitual. El directorio además tiene la función de actuar como directorio raíz de la página web y alberga el resto de directorios relacionados con el proyecto.

Así dentro de SCHEDGRID podremos encontrar los siguientes directorios:

- El directorio **exe** contendrá los programas ejecutables, el usuario no tendrá acceso directo a ellos pero la interfície web los utilizará para llevar a cabo algunas de sus funciones. Dentro de exe podremos encontrar los ejecutables de los tres tipos de heurísticas tratadas, además de un fichero de información que nos valdrá de índice del directorio y como breve guía de cómo utilizar los diferentes programas disponibles.

- El directorio **ico** contendrá los ficheros con información gráfica de la interfície web. Entre otras imágenes podemos encontrar el icono de la FIB o la imagen que se muestra en la parte superior de la página web. Si el administrador de la página decidiera cambiar alguna de las imágenes mostradas solo tendría que sustituir la imagen no deseada por la nueva manteniendo el mismo nombre.

- El directorio **instances** contendrá las instancias que se ofrecerán al usuario en caso de que el usuario no quiera aportar su propia instancia. Las instancias que se encuentran en este directorio han sido probadas previamente por el administrador de la página.

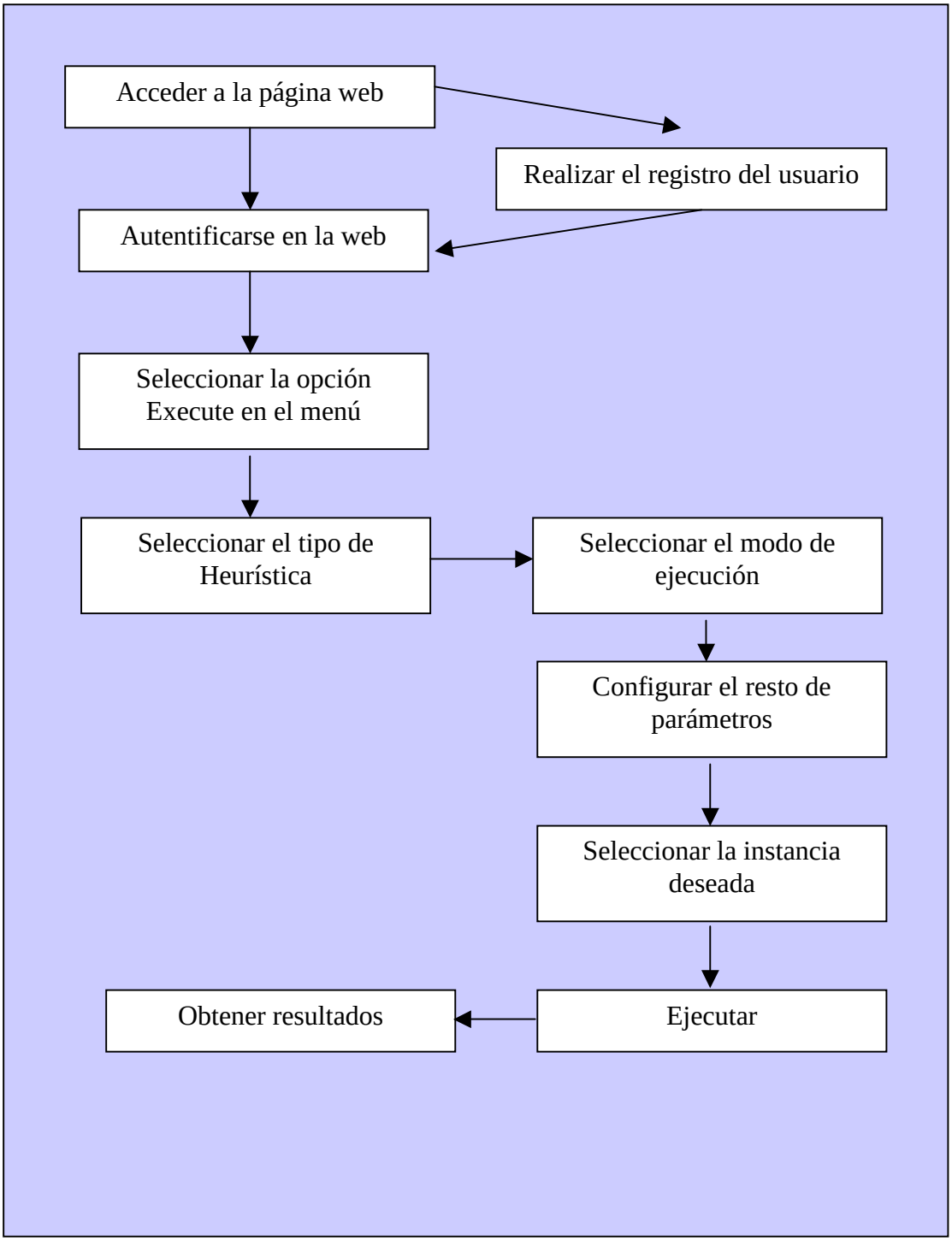
- El directorio **jobs** contendrá una carpeta para cada usuario, además de una carpeta llamada TryIt. En estas carpetas se guardarán las ejecuciones realizadas por cada usuario, tanto el fichero de entrada a los ejecutables como el fichero de salida, a excepción de en la carpeta TryIt donde solo se guardarán los ficheros que contienen las matrices ETC de entrada a los ejecutables.

- El directorio **lib** contendrá los ficheros necesarios para cargar información en la página web y ayudar en medida de lo posible a la navegación de los usuarios por la interfície. Los ficheros que encontramos en este directorio son ficheros que contienen las operaciones que serán llamadas desde el directorio raíz.

- Finalmente encontramos el directorio **proposals**, donde encontramos las instancias propuestas por los usuarios, de tipo investigador, para convertirlas en instancias estándares o públicas. Una instancia pasa a ser pública después de que el administrador de la página haya evaluado su funcionamiento y haya comprobado que es una instancia válida.

4.2. Flujo de trabajo

Tal y como se ha comentado anteriormente, en este apartado se quiere mostrar de manera gráfica el camino que han de seguir los usuarios para realizar algunas operaciones dentro de la interfície web. Debido a que la página web da la posibilidad de ejecutar una gran variedad de flujos de trabajo se ha optado por mostrar sólo uno de ellos, ya que el resto son en cierta manera semejantes. El flujo que mostraremos pertenece al flujo seguido para la ejecución de un programa de heurística desde la interfície web.



El usuario primero tiene que acceder a la página web, donde se mostrará la pantalla de bienvenida. En caso de que el usuario no este registrado (solo debe registrarse una vez) deberá realizar su registro, para ello deberá dirigirse a la parte derecha de la página y pulsar sobre "New user" y completar los campos solicitados para dar de alta su cuenta de usuario.

Una vez registrado el usuario, podemos seguir con el flujo de trabajo habitual, para ello deberemos ir desde la página de bienvenida de nuevo al menú de la derecha y seleccionar "Login". Nos aparecerá una nueva pantalla donde debemos identificar el usuario creado anteriormente o el que ya teníamos creado de otro acceso a la página.

Una vez logueados, tendremos ante nosotros una pantalla muy similar a la pantalla de bienvenida pero con algunas cambios en sus menús. Ahora para ejecutar un programa de scheduling deberemos ir al menú de la izquierda y seleccionar "Execute".

Una vez seleccionada la opción "Execute" empezaremos a configurar el tipo de ejecución que vamos a realizar, así la primera pantalla que nos muestra la web es la de selección de tipo de heurística. Seleccionaremos la heurística por defecto "Heurísticas Ad-hoc" y continuamos con la configuración.

En la siguiente pantalla deberemos seleccionar el modo de ejecución de la heurística en este caso, igual que en el anterior, seleccionaremos la opción por defecto "Batch" y continuaremos con la configuración.

En la pantalla que se nos muestra a continuación tendremos la posibilidad de configurar tres opciones de la configuración, el primero seleccionar la heurística que se ejecutará, las diferencias entre una y otras han sido explicadas en el apartado 2 de esta misma memoria. Lo siguiente a configurar son la introducción de los parámetros, los cuales decidiremos introducir mediante una configuración existente. Por último deberemos decidir si queremos que la solución a nuestro problema nos sea enviada a nuestro correo o simplemente queremos ver un resumen de ésta por pantalla. Seleccionaremos la opción de enviar por correo[2].

En la siguiente pantalla podremos seleccionar de un desplegable la instancia que vamos a ejecutar.

Por último el usuario solo tendrá que ir a la dirección de correo con la cual realizó el registro de usuario y leer los resultados obtenidos de la ejecución.

[2] La solución será enviada a la dirección de mail con la cual el usuario se dio de alta.

4.3. Diseño Gráfico

Todas las secciones de la página a las que podemos acceder en la interfície web mantienen una misma estructura gráfica, a excepción de la página de login. En todos las secciones podemos ver una distribución basada en tres partes: La primera es el menú de la izquierda, el cual va cambiando a medida que se navega por la página. En el apartado público mantiene siempre su mismo aspecto y cuando nos logueamos podemos ver como dependiendo del tipo de usuario muestra unas opciones o otros. La segunda parte es la parte central, donde se muestra la información para interactuar con el usuario. Por último la tercera parte es el menú de la derecha, el cual nos permite loguearnos y dar de alta nuevos usuarios.

Como podemos observar hay una cuarta parte, la cual no he mencionado, que encontramos situada en la parte superior de la página. Esta parte esta formada por una imagen la cual tiene como único propósito que la parte de interacción del usuario con la página web quedé centrada en el centro de la pantalla del usuario.

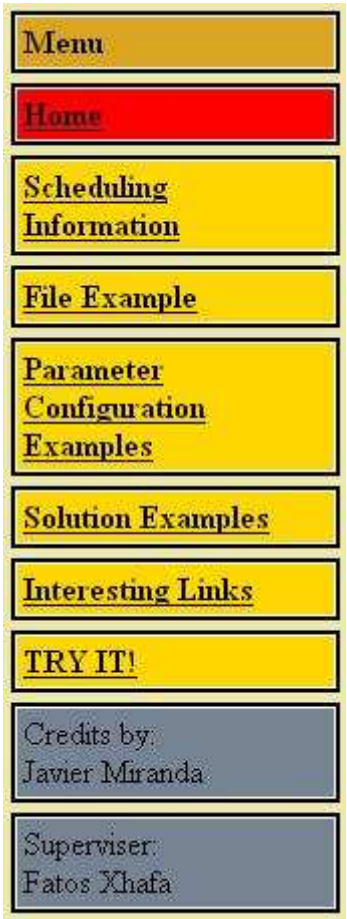
<table>
<tr><td colspan="3">Imagen superior</td></tr>
<tr><td>Menú Izquierdo</td><td>Parte Central</td><td>Menú Derecho</td></tr>
</table>

Me gustaría destacar que cada una de las partes se encuentran en diferentes ficheros, de forma que si alguna vez se quisiera realizar algún cambio sobre el contenido de alguna de ellas tan solo tendríamos que modificar el fichero que contiene la información de la parte selecciona. De esta forma evitamos tener que realizar un sin fin de cambios en los archivos de la página web.

4.3.1. Menú Izquierdo

El menú izquierdo informa sobre la navegación de la web mostrando un contenido dinámico a la vez que se va navegando. En color rojo se muestra la sección en la que estamos actualmente, mientras que si nos ponemos sobre otra sección con el ratón podemos observar como el color del botón cambia a marrón. Al final del menú podemos encontrar los créditos de quien ha realizado la página y del supervisor que ha tenido.

Podemos encontrar cuatro versiones diferentes del menú de la izquierda: la pública, la de invitado, la de investigador y por último la de administrador. Las tres últimas se mostrarán dependiendo del tipo de usuario que se logueé en la página de login.

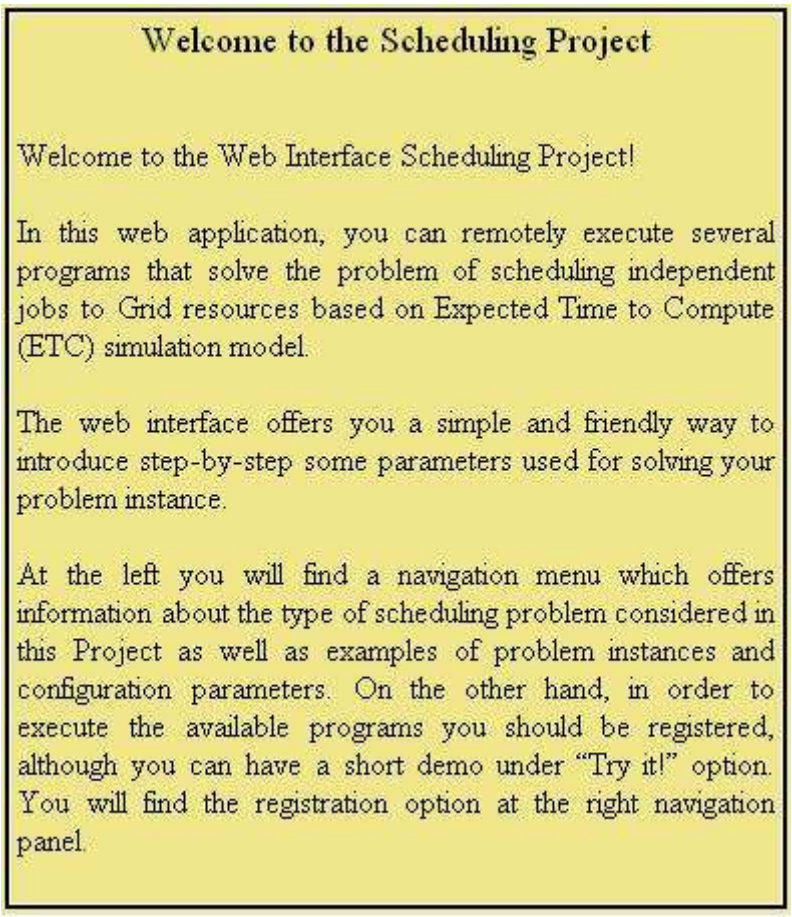


Menú Público

Menú Administrador

4.3.2. Parte Central

En la parte central se mostrará el contenido principal de la sección seleccionada, tanto en el menú de la izquierda como en el menú de la derecha. Así por ejemplo cuando accedemos a la página por primera vez, la parte central nos mostrará el siguiente mensaje de bienvenida:



4.3.3. Menú Derecho

El menú derecho, al igual que el izquierdo informa sobre la navegación de la web mostrando un contenido dinámico a la vez que se va navegando. El color rojo en este caso resalta el botón más importante del menú, el botón de LOGIN. Si nos situamos sobre algún botón con el ratón podemos observar como el color del botón cambia a marrón a excepción del botón de LOGIN, el cual cambia a gris. Al final del menú podemos encontrar tres imágenes, las cuales a su vez son links a páginas que tienen algún tipo de relación con el proyecto: FIB, UPC, Departamento del LSI.

De este menú solo encontramos dos versiones diferentes: la pública y la de usuario registrado. En éste caso el menú mantiene una parte fija y otra dinámica, la parte fija es la parte donde se muestran los links-imágenes y la parte dinámica es donde se muestran los botones de “Login”, “New User” y “Forgotten the Password”.

Menú Público

Menú Usuario

4.4. Estructura del apartado público de la web

El apartado público de la página web esta dividido en diez secciones. Cuando se introduce la URL del website se carga la HomePage o página de Bienvenida, que contiene un mensaje de bienvenida para todo usuario que acceda a la página. A partir de esta página se puede acceder a cualquiera de las otras nueve secciones, las cuales se mencionan a continuación:

Esquema Global

- Home Page
- Scheduling Problem Information
- File Examples
- Parameter Configuration Examples
- Solution Examples
- Interesting Links
- Try it!
- Login Page
- New User : Registro
- Forgotten Your Password

4.4.1. Home Page

Fichero: index.php

Esta sección es la primera que el usuario visualiza cuando accede a la web. Es la encargada de dar la bienvenida al usuario y dar una pequeña introducción sobre el funcionamiento de la web.

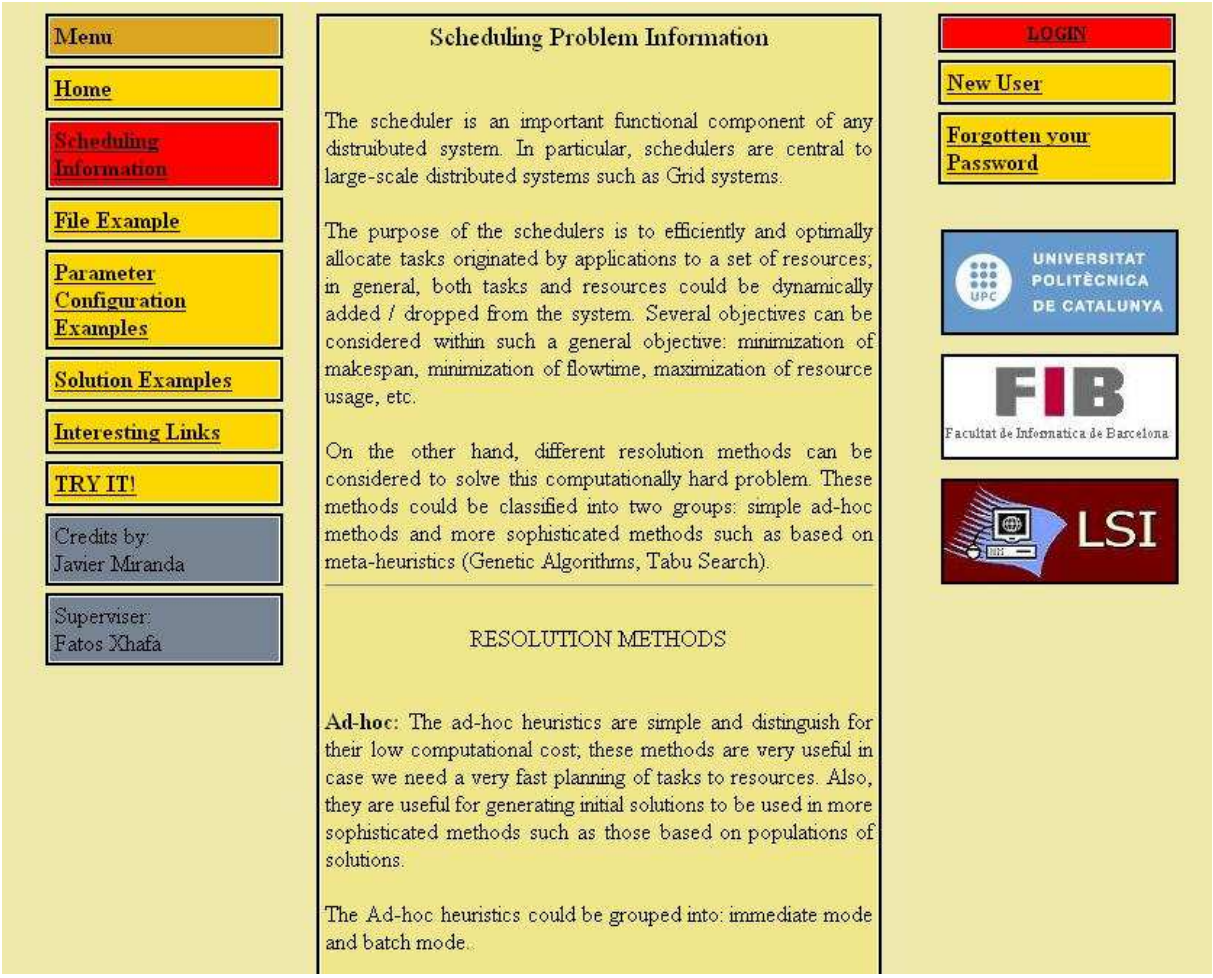
4.4.2. Scheduling Problem Information

Fichero: sched_information.php

Esta sección es la encargada de explicar en que consiste un problema de scheduling y porque es importante solucionarlo. Por otro lado también obtenemos información acerca de los diferentes métodos de resolución con los que vamos a poder tratar nuestras instancias: Ad-hoc, Genetic algorithm y Tabu Search.

En esta sección también podemos encontrar información sobre el tipo de instancias que acepta el programa, donde se explica en que consiste una matriz ETC. Por último se da una breve explicación de los distintos parámetros que la web nos muestra después de realizar una ejecución: Makespan, Utilization, Flowtime y Matching Proximity.

A continuación se muestra una figura con parte de su contenido:

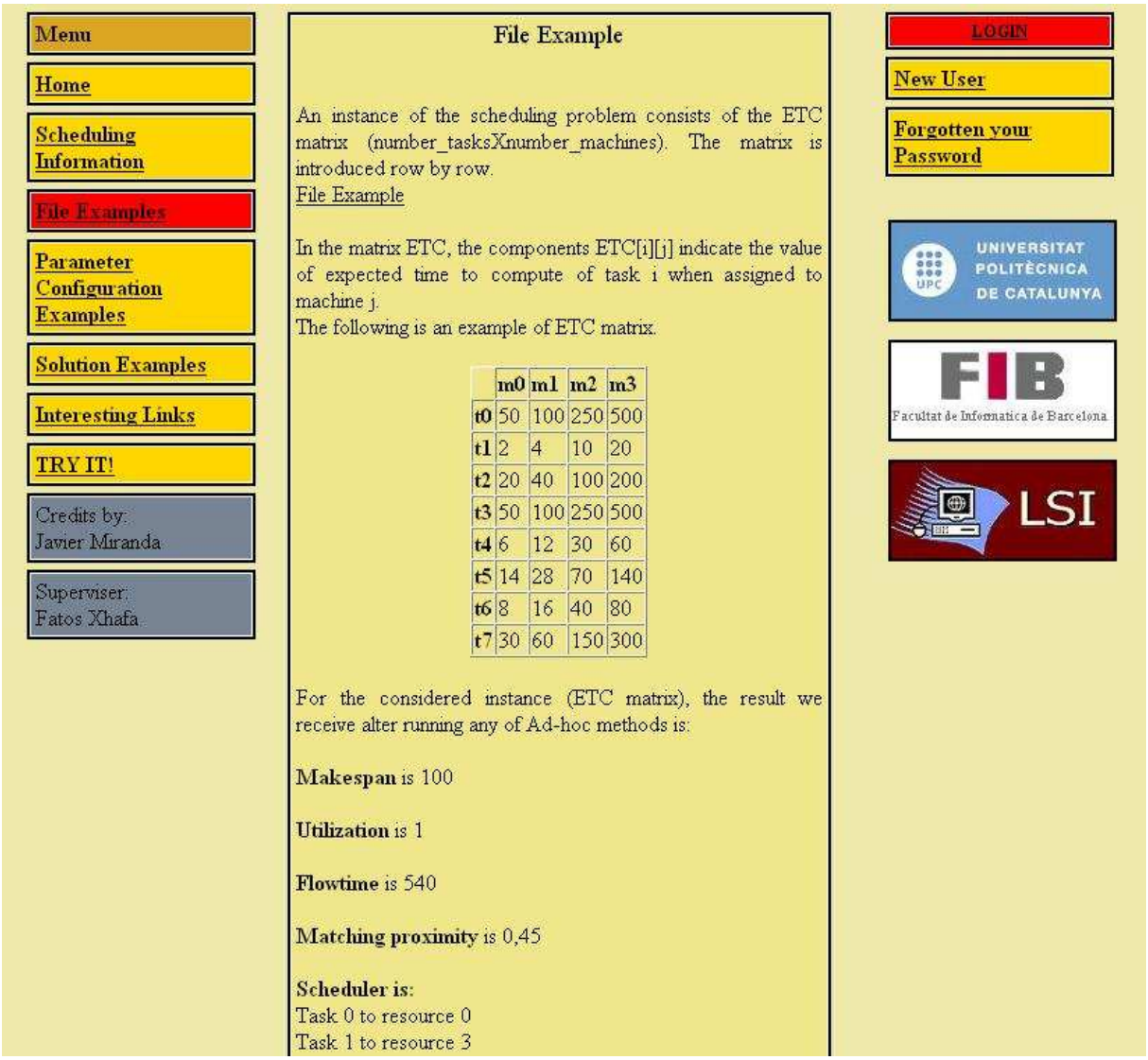


4.4.3. File Examples

Fichero: filesExample.php

En esta sección se nos explica que contenido debe llevar una instancia apta para la ejecución de uno de los programas ofrecidos en la web. Al igual que en la sección anterior se explica en que consiste una matriz ETC y se da un ejemplo completo de una matriz de ocho tareas y cuatro máquinas con su respectiva solución.

Por otro lado hay que destacar que mediante un enlace podemos acceder a una instancia de ejemplo donde podemos ver de que manera y con que formato se deben encontrar los datos en una instancia válida para la ejecución.

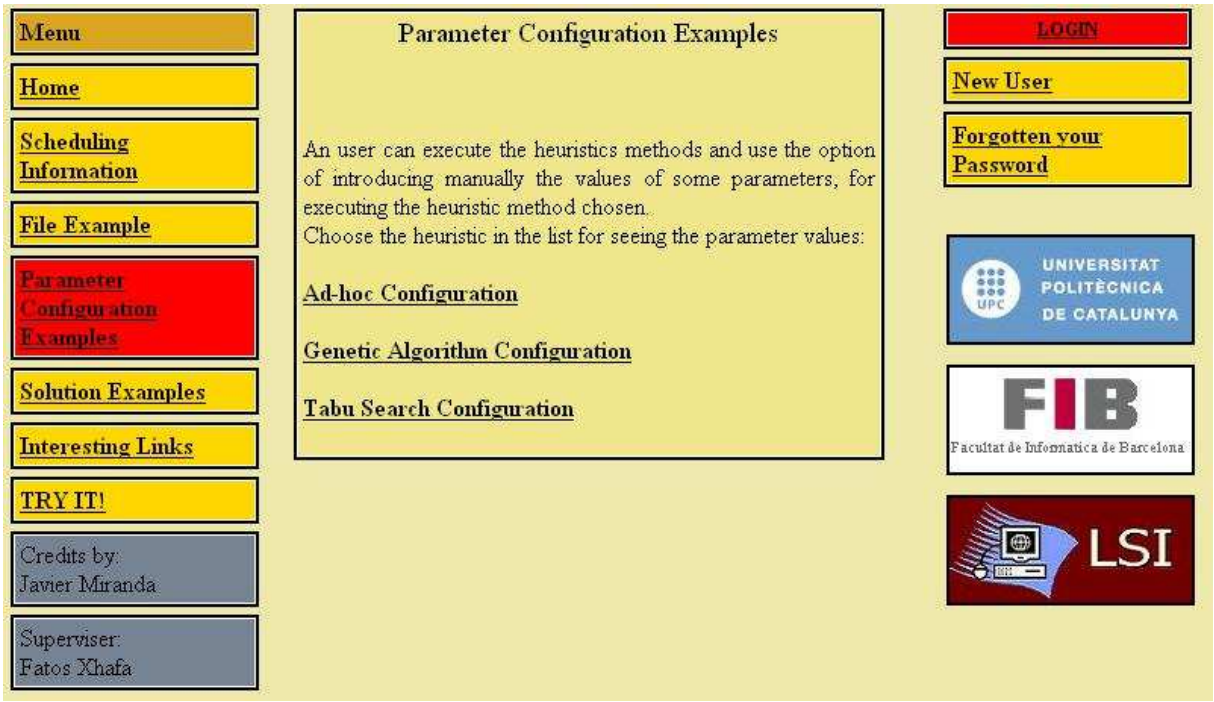


4.4.4. Parameter Configuration Examples

Fichero: parameters.php

En esta sección se muestra una lista de los métodos de resolución que podemos utilizar para dar solución a nuestros problemas de Scheduling. Seleccionando uno de estos métodos obtendremos la información necesaria sobre los parámetros que tendremos que configurar para ejecutar dichos métodos.

En la figura que se muestra a continuación se muestra la lista de métodos de resolución:



4.4.5. Solution Examples

Fichero: answer.php

En esta sección se explica las diferentes maneras en que se pueden conseguir los resultados de una ejecución: visualizando los resultados por pantalla, en este caso no se mostrará información sobre que tarea va a ejecutarse en cada máquina, o visualizar los resultados en la cuenta de correo.

4.4.6. Interesting Links

Fichero: links.php

Esta sección muestra al usuario información sobre links que puede resultar interesante visitar ya que mantienen cierta relación con la web realizada para el proyecto.

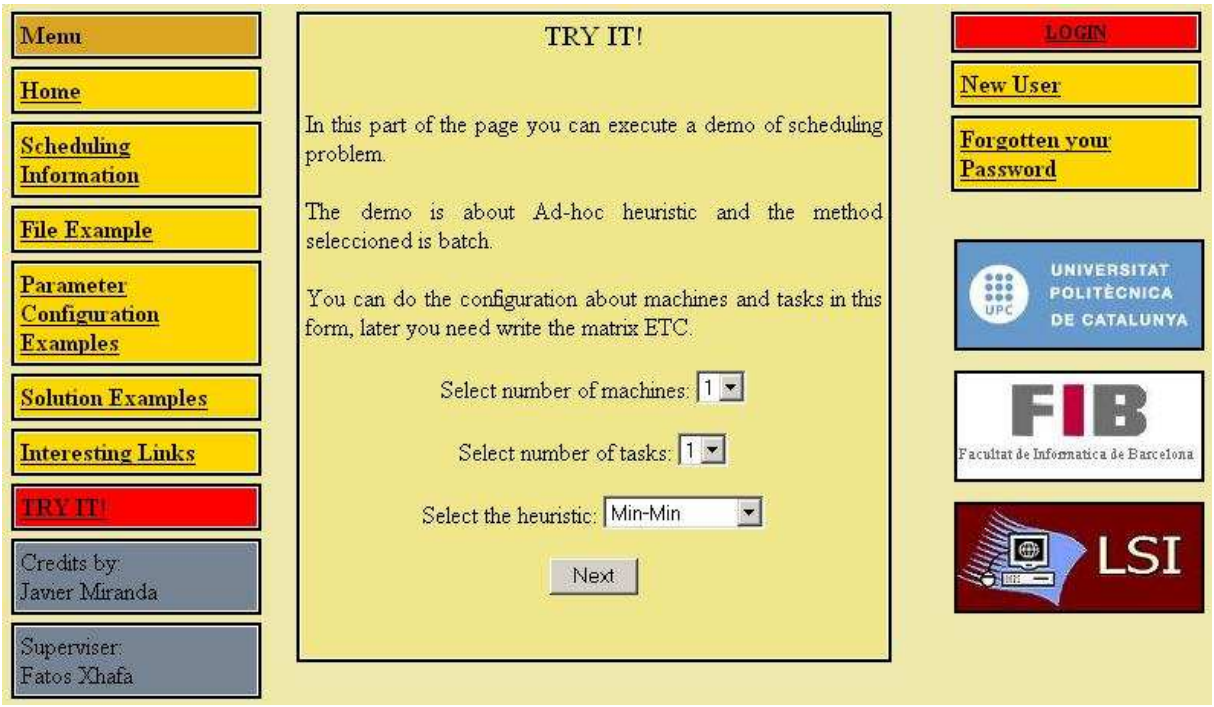
4.4.7. Try It

Fichero: try_it.php

Esta sección permite al usuario jugar con una pequeña demo la cual permite al usuario ver lo que se ofrece en la web. El fin de esta demo es mostrar el producto al cual se puede acceder si un usuario se registra. La demo como su propio nombre indica es simplemente una demostración, así que se han fijado algunos parámetros de ejecución y se han limitado el numero de máquinas a cuatro y el número de tareas también se han limitado a cuatro.

Así en esta sección se pide al usuario que seleccione el número de máquinas y el número de tareas entre un número comprendido entre uno y cuatro. Después el usuario decide que tipo de heurística de tipo Ad-hoc y bach ejecuta.

Esta captura muestra los parámetros configurables de esta sección:



Una vez configurado estos parámetros la página pedirá que se introduzcan en una matriz los tiempos que tarda cada tarea en ejecutarse en cada una de las máquina. Una vez introducido los datos se mostrará por pantalla el resultado de la ejecución.

4.4.8. Login Page

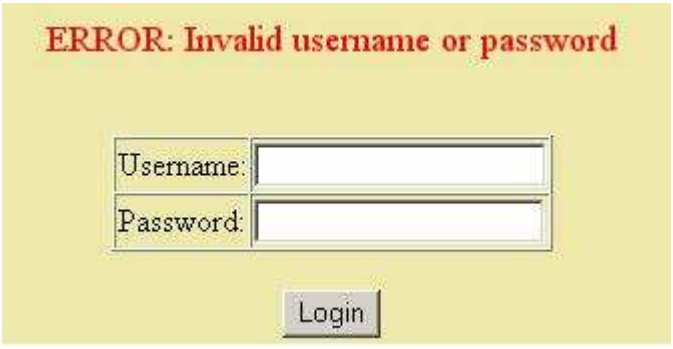
Fichero: login.php

Esta sección es una de las más importantes ya que consiste en la identificación y verificación de los usuarios. Es la puerta de entrada a la parte privada de la página.

En caso de que el usuario no haga correctamente el login aparecerá un mensaje de error, ya sea porque el usuario no existe o porque ha introducido mal la contraseña. Además la sección permite acceder de forma directa a las últimas dos secciones de este apartado: New User y Forgotten your Password.

A continuación se muestran unas figuras sobre el funcionamiento de esta parte de la aplicación:

Si el usuario realiza el login de forma incorrecta aparece una pantalla mostrando el siguiente mensaje de error:



Por último comentar que el control de usuarios se realiza en la base de datos. El SGBD es el encargado de garantizar que no existen dos usuarios con un mismo username.

4.4.9. New User

Fichero: register.php

Esta sección es la encargada de dar de alta nuevos usuarios. Los futuros usuarios deberán rellenar un formulario en el cual si solo se rellenan los datos básicos se realizará un alta de tipo invitado, en cambio si se rellenan todos los datos se realizará un alta de tipo investigador.

El formulario obliga a los futuros usuarios a introducir dos veces la clave, evitando de esta manera que se produzca un error en la introducción de la misma. También se realiza una comprobación en la dirección de e-mail comprobando si el dato introducido puede ser una posible dirección de correo electrónico. Por último, destacar que cuando se crea un usuario, independientemente del tipo que sea, simultáneamente se crea una carpeta dentro del directorio **jobs** con el nombre del usuario para guardar sus futuras ejecuciones.

4.4.10. Forgotten Your Password

Fichero: rememberpass.php

Esta sección es la encargada de recordar a nuestros usuarios su contraseña en caso de que la hayan olvidado. Para que la web recuerde la contraseña solo se debe introducir el username con el que el usuario se haya dado de alta. Inmediatamente la web enviará un e-mail a la dirección de correo dada durante el registro con la contraseña.

4.5. Estructura del apartado privado de la web

El apartado privado de la página web esta dividido en dos partes bien diferenciadas:

En primer lugar tenemos la parte del **administrador**, quien dispone de nueve secciones diferentes para moverse a través de la página una vez ha realizado el login. En segundo lugar tenemos la parte del **invitado y el investigador**, entre las que no haremos diferencia ya que ambas partes son muy similares. Estos últimos disponen de seis secciones diferentes.

Cuando un usuario realiza el login éste accede a su parte privada de la página. Independientemente del tipo de usuario que se logueé, el website cargará el Home o página de bienvenida de usuario registrado. A partir de esta página se puede acceder a cualquiera de las otras secciones.

4.5.1. Parte privada del Administrador

Esquema Global del Administrador

- Home
- Information about Users
- Configuration User
- Delete Users
- News
- To see proposals
- Instances
- My Account
- Statistics

4.5.1.1. Home

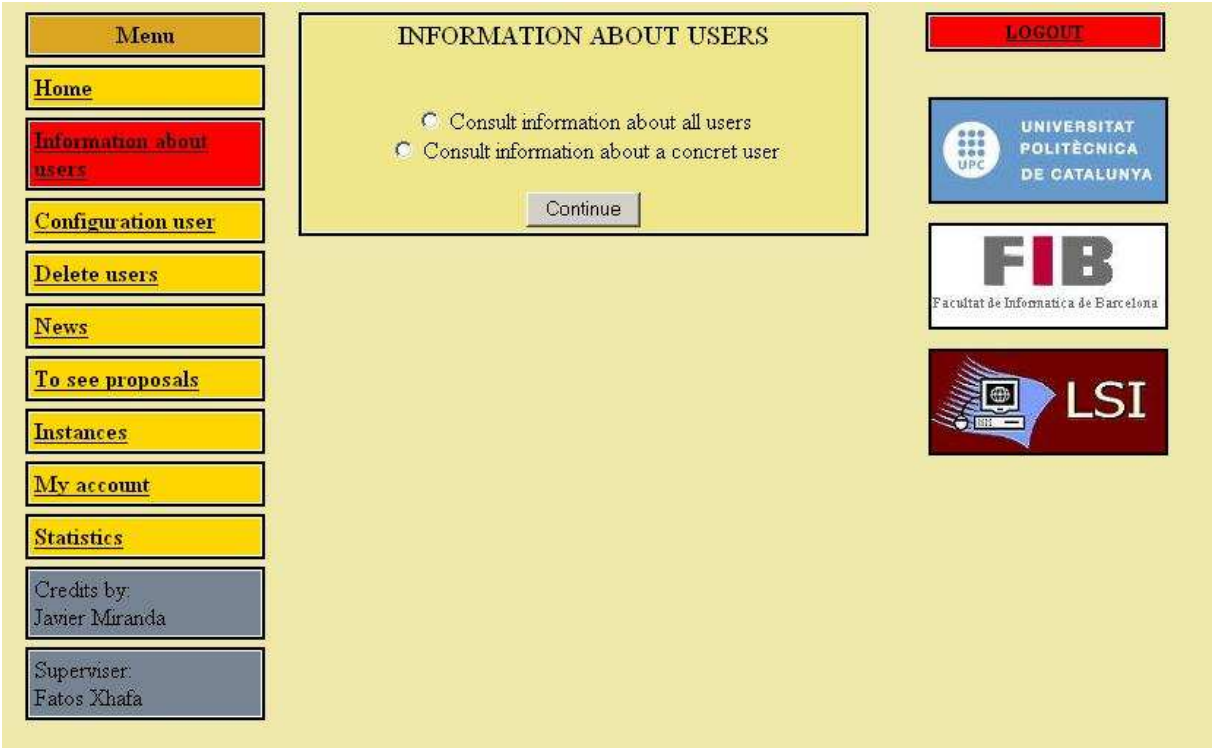
Fichero: homeAdmi.php

Esta sección es la encargada de dar la bienvenida al usuario y de explicarle alguna de las acciones que puede realizar en la parte privada de la página. En este caso explicará que puede realizar la gestión de usuarios y de noticias desde el menú de la izquierda y que puede mirar las instancias propuestas por los invitados además de las estadísticas generales de la web.

4.5.1.2. Information about Users

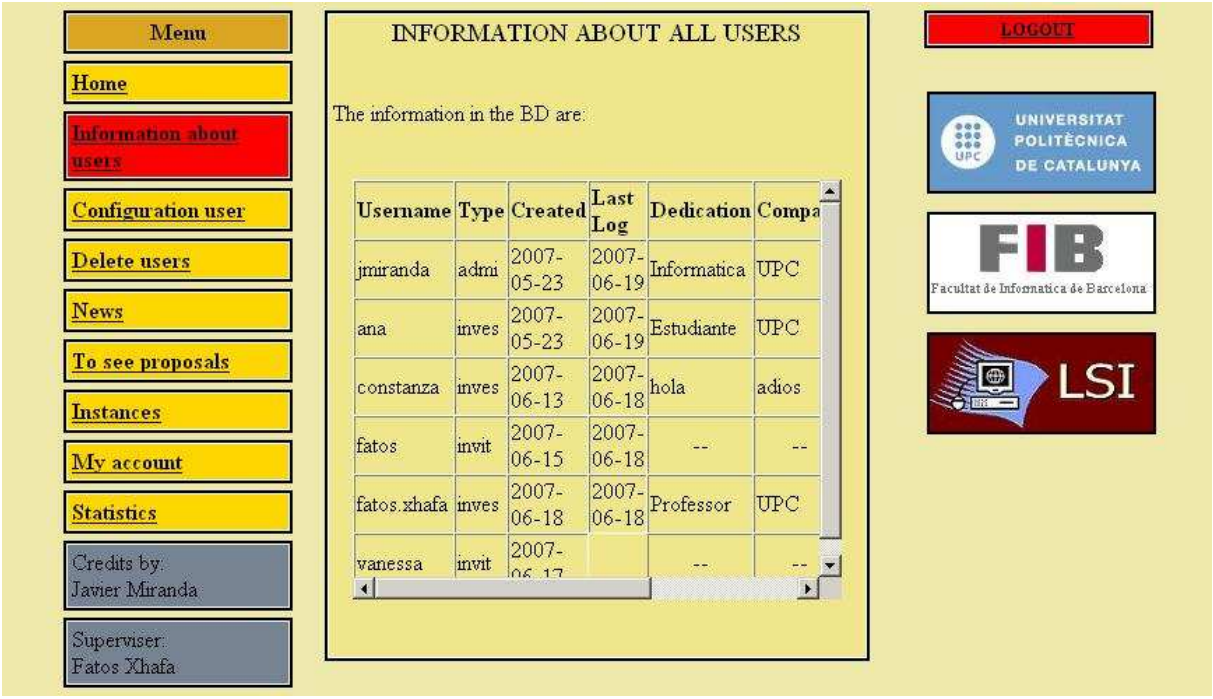
Fichero: infouser.php

Esta sección se encarga de mostrar al administrador la información que los usuarios han introducido al hacer el registro. El administrador tendrá la opción de elegir ver la información de un solo usuario o de todos los usuarios que están registrados en la página, para ello se muestra la siguiente pantalla de selección:



En caso de que el usuario seleccione ver la información sobre un usuario concreto, en la parte central de la página se le mostrará un desplegable en donde podrá elegir el usuario deseado. Una vez seleccionado se le mostrará la información del usuario.

Por otro lado, si el administrador decide ver la información de todos los usuarios la interfaz web lo que mostrará será una tabla donde los campos del registro aparecerán como columnas y los usuarios como filas.



Destacar que entre la información mostrada el único campo del registro que no se muestra es la contraseña del usuario.

4.5.1.3. Configuration User

Fichero: formuser.php

Esta sección permite al administrador configurar algunos parámetros de las ejecuciones de los invitados e investigadores. Al administrador inicialmente se le mostrará el estado actual de la configuración y si él lo viese conveniente podría realizar los cambios que considerará oportunos.

Las opciones configurables son: el número máximo de ejecuciones y el tiempo máximo que puede tardar un algoritmo genético o de tabu search resolviendo un problema.

Además podremos decidir que tipo de ejecución van a realizar los usuarios y el número de últimas ejecuciones que podrán ver los usuarios en el apartado: “I want to see my last executions”.

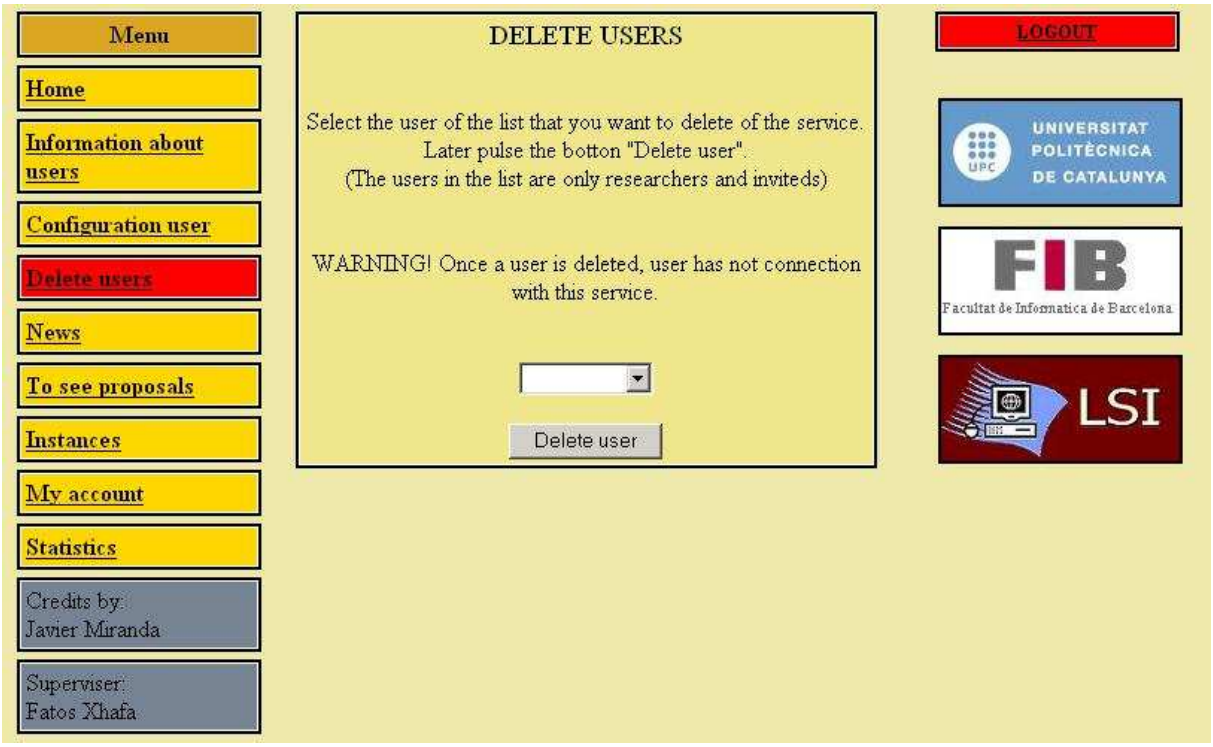
4.5.1.4. Delete users

Fichero: delete.php

Esta sección permite al administrador borrar cualquier usuario invitado o investigador de la web. Cuando el administrador accede a la sección se le muestra un texto explicando las consecuencias que sufrirá el usuario una vez borrado.

El administrador deberá seleccionar de un desplegable el usuario que desea eliminar del servicio, como seguridad el desplegable no mostrará ningún nombre por defecto, así se consigue que nunca se borre un usuario por error ya que el administrador debe equivocarse dos o incluso tres veces para borrar un usuario de forma no deseada (Deberá ir a la sección delete user, una vez allí seleccionar el usuario y después pulsar el botón “Delete user”).

Una vez el administrador haya seleccionada un usuario y pulsado sobre el botón inferior de la pantalla se mostrará un mensaje de confirmación o de error dependiendo si el usuario ha podido ser borrado o no.



Como último para esta sección, me gustaría comentar que en el listado de usuarios disponibles para eliminar del servicio nunca se encontrarán los nombres de los administradores de la página.

La eliminación de un administrador no se podrá hacer desde la interfaz web, como eliminar un administrador se explica en el manual de administrador que se encuentra en los anexos, al final de esta misma memoria.

4.5.1.5. News

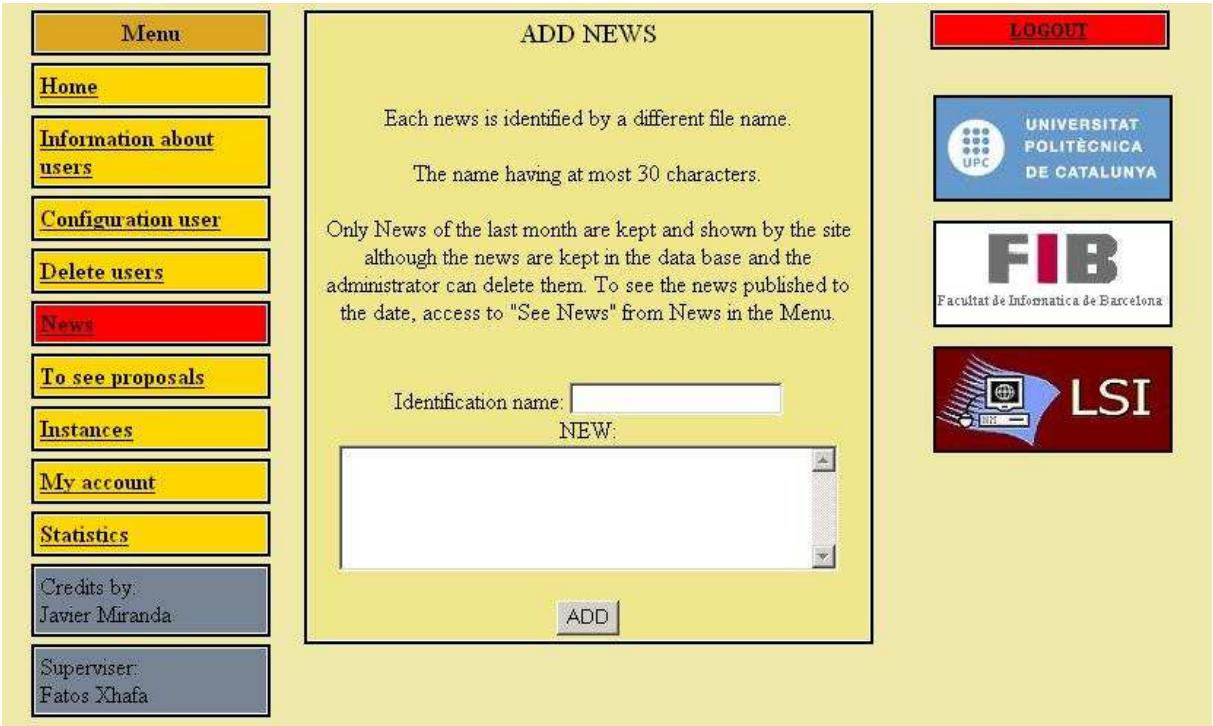
Fichero: newsAdmi.php

Esta sección está separada en tres apartados: see, add y delete news.

El primero de estos apartados permite al administrador visualizar todas las noticias publicadas hasta el momento por el mismo o por otro administrador, ya que los únicos que pueden publicar noticias son los administradores de la web. Las noticias serán mostradas ordenadas por fechas, las más recientes aparecerán más arriba en la lista y las más antiguas más abajo.

El segundo apartado permitirá al administrador añadir una nueva noticia a las existentes. Cada noticia debe ir acompañada de un nombre identificativo el cual será mostrado cuando se muestre la noticia.

A continuación se muestra la pantalla para la introducción de nuevas noticias:



En la pantalla se explica que las noticias deben tener un nombre identificativo como he comentado anteriormente y además se añade que las noticias serán visibles por los investigadores/invitados durante un mes desde su fecha de publicación

Por último encontramos el apartado de eliminación de noticias existentes, en este apartado deberemos seleccionar de un desplegable el nombre identificativo de la noticia que deseamos borrar y pulsar sobre “Delete”. Al igual que con el borrado de usuarios, el desplegable inicialmente no muestra seleccionada ninguna noticia, así evitamos borrar noticias de forma no deseada.

4.5.1.6. To see proposals

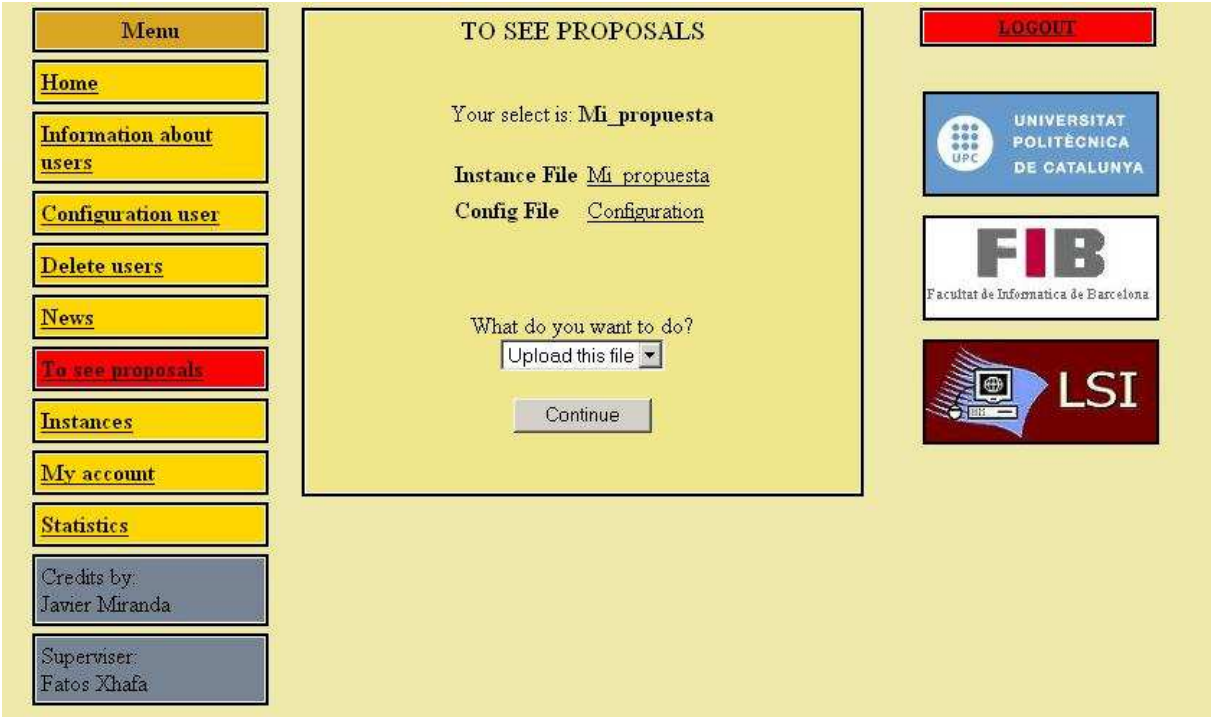
Fichero: propAdmi.php

En esta sección el administrador podrá ver si algún usuario investigador a propuesto alguna instancia para convertirla en instancia pública. En caso de que haya más de una propuesta disponible se mostrarán los nombres identificativos de las propuestas, para que el administrador seleccione una de ellas.

Una vez seleccionada la propuesta el administrador verá en la parte central de la pantalla información sobre la instancia: número de máquinas y número de tareas.

Si el administrador cree que conviene publicar la instancia deberá realizar las pruebas pertinentes para saber si se trata de una instancia válida o no. Una vez haya realizado las pruebas, según lo vea conveniente el administrador podrá poner la instancia a disposición pública o eliminarla de la sección de instancias propuestas.

A continuación se muestra la pantalla que puede ver el administrador una vez ha seleccionado ver una propuesta:



4.5.1.7 Instances

Fichero: instances.php

Mediante esta sección el administrador podrá gestionar las instancias que se encuentran en la aplicación.

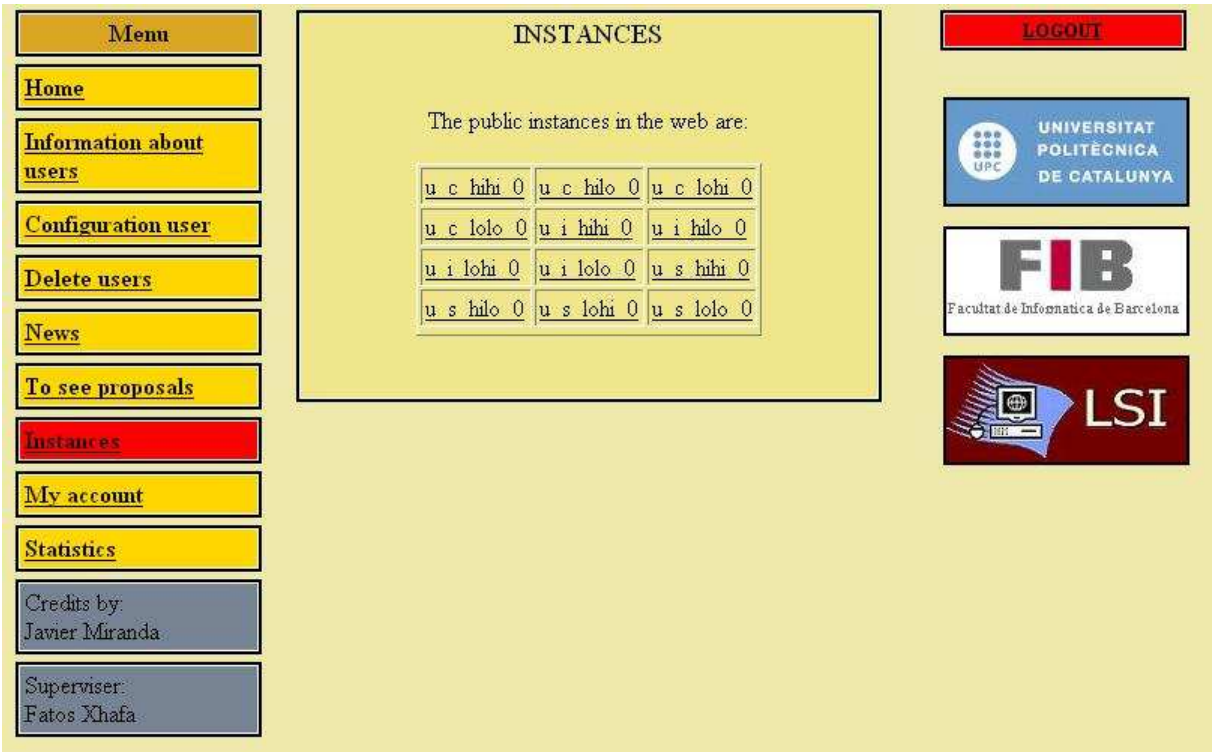
Inicialmente se le mostrarán tres opciones: ver las instancias públicas de la aplicación, añadir una nueva instancia o borrar una instancia.

En la sección de añadir una nueva instancia el usuario deberá poner un nombre a la instancia y seleccionar la instancia de su propio sistema de ficheros. Si el nombre es válido la instancia será subida al servidor de la aplicación web. En caso contrario se mostrará un mensaje de error.

Para eliminar una instancia, el administrador deberá seleccionar la instancia de un desplegable en el cual se mostrarán todas las instancias públicas que se encuentran en la aplicación. Después deberá pulsar sobre el botón: “Delete”

Si en caso contrario el administrador decidiera ver las instancias públicas en la aplicación, ésta mostraría una tabla con todas las instancias públicas. El usuario pulsando sobre el nombre de la instancia podrá ver su contenido.

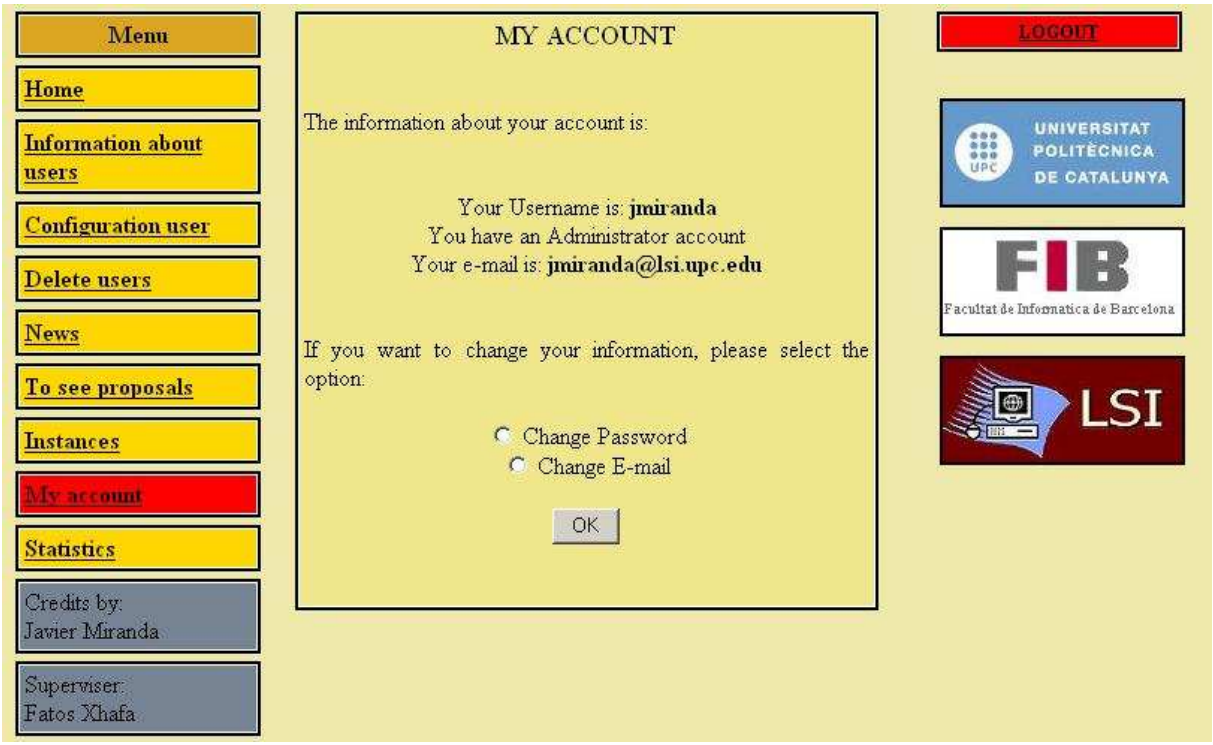
A continuación se muestra la pantalla donde el usuario podrá ver las instancias publicadas hasta el momento:



4.5.1.8 My Account

Fichero: MyAccount.php

En esta sección el usuario podrá gestionar parte de su cuenta de usuario. En una primera pantalla se muestra la información sobre el usuario y se da la posibilidad de cambiar alguno de los datos introducidos en el registro:



Si opta por cambiar el password, la web le llevará a una pantalla donde deberá escribir el password actual de usuario y el nuevo password obligándole a repetir el nuevo password para evitar posibles errores en el mecanografiado del mismo.

Por otro lado si decide cambiar el e-mail no deberá introducir ningún dato identificador, simplemente introducir el nuevo mail y éste substituirá al antiguo.

4.5.1.9. Statistics

Fichero: estadisticas.php

Esta sección contiene las estadísticas generales del la web. El administrador podrá ver entre otras cosas el número de ejecuciones que se han realizado en la página y el número de usuarios que están conectados a la aplicación actualmente.

El administrador también tendrá la posibilidad de establecer un contador parcial de ejecuciones, el cual se reinicia pulsando sobre el botón: “Reset this number with present date”

En la parte final de las estadísticas podemos encontrar un apartado dedicado al número de usuarios registrados en la aplicación, los usuarios están diferenciados por tipo, así que en todo momento sabremos el número de invitados e investigadores de los que dispone nuestra aplicación.

A continuación se muestra una vista parcial de la pantalla de estadísticas:

4.5.2. Parte privada del Invitado/Investigador

Esquema Global del Invitado/Investigador

- Home
- News
- Execute
- I want to see my last executions
- My Account
- Propose an Instance

4.5.2.1. Home

Fichero: homeInvestigador.php y homeInvitado.php

Esta sección es la encargada de dar la bienvenida al usuario y de explicarle alguna de las acciones que puede realizar en la parte privada de la página. En este caso explicará entre otras cosas que puede ejecutar los programas de scheduling, que puede gestionar su propia cuenta de usuario y ver las noticias publicadas por el webmaster.

4.5.2.2. News

Fichero: newsInv.php

Esta sección esta dedicada a la visualización de las noticias publicadas por el administrador. Los usuarios invitados tanto como los usuarios investigadores podrán leer las noticias que se han publicado en el último mes.

Las noticias serán mostradas ordenadas por fechas, las más recientes ocuparán las posiciones más altas de la lista, mientras que las más antiguas ocuparan las más bajas.

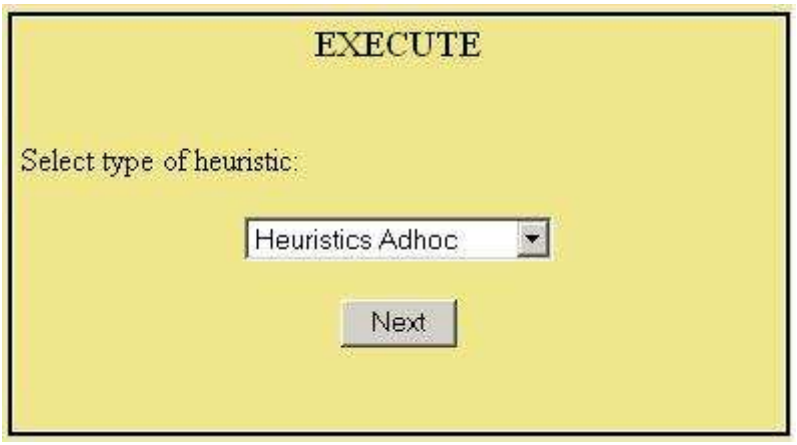
El aspecto de todas las noticias es el mismo: inicialmente encontramos el nombre identificativo de la noticia, seguido de la fecha en que fue publicada y acompañada del texto con el contenido de la noticia.

4.5.2.3. Execute

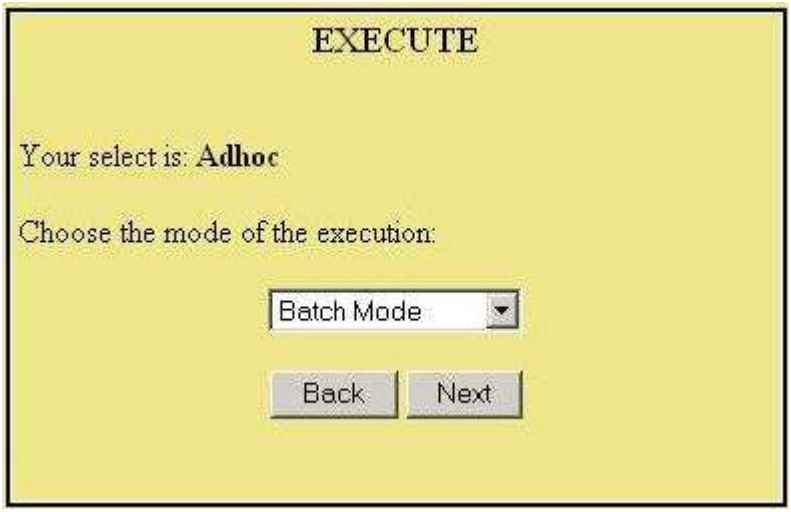
Fichero: execute.php

Podríamos decir que esta es la sección más importante de la web, ya que el proyecto consiste en realizar una interfaz web para la ejecución de programas de scheduling. Así en esta sección, encontramos los diferentes programas que nos ofrece la web para ejecutar nuestros problemas.

En la primera pantalla nos muestra un desplegable con todos los métodos de heurística entre los que podemos elegir para solucionar nuestro problema:

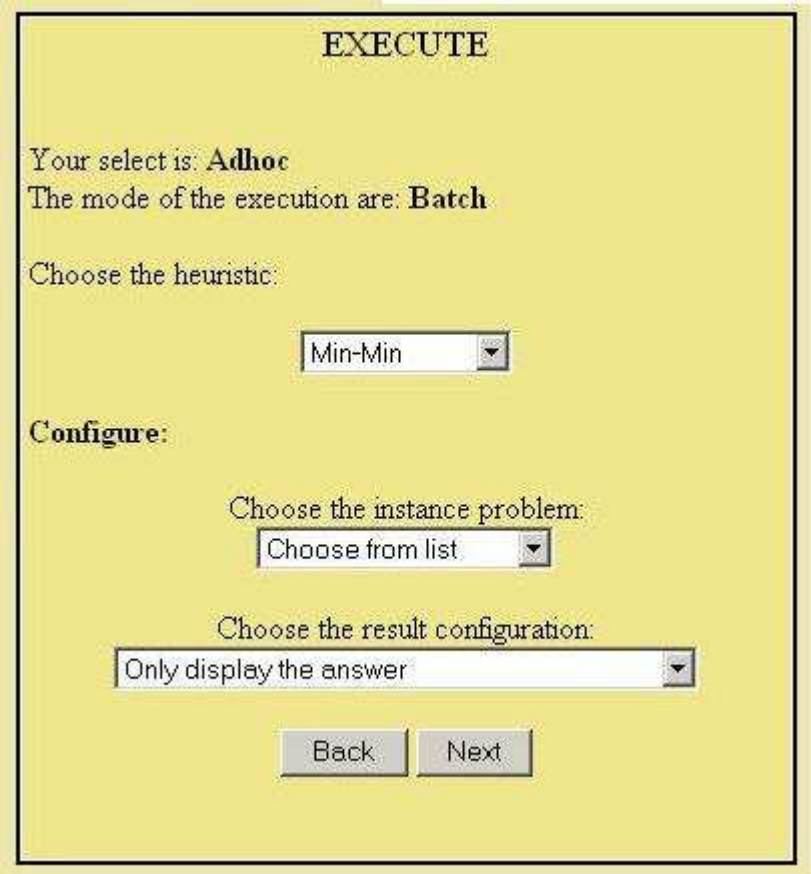


En la siguiente pantalla (si elegimos el método Ad-hoc, por ejemplo) tendremos la oportunidad de elegir el tipo de modo de ejecución:

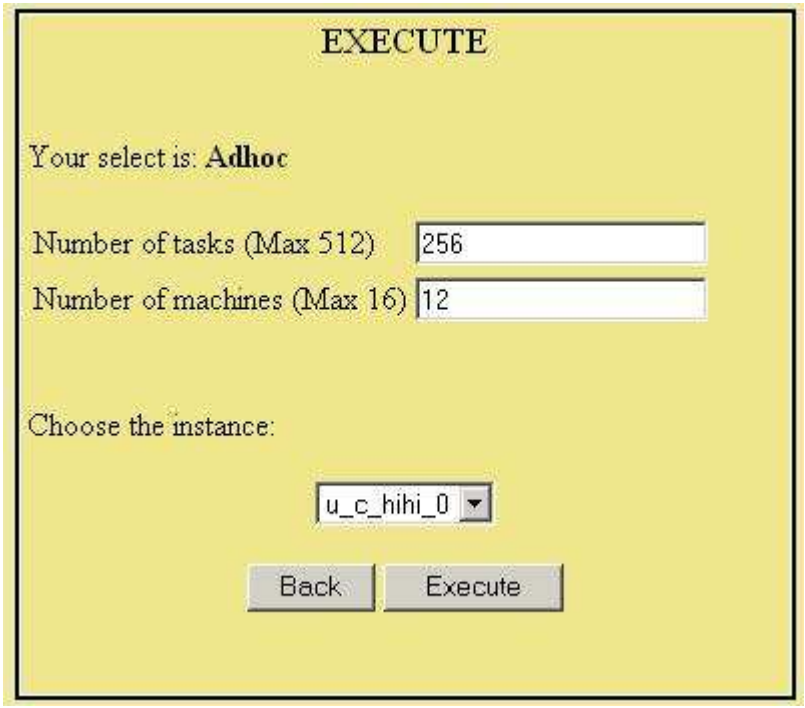


En la tercera pantalla es donde acabaremos de configurar los parámetros para la ejecución, seleccionaremos el tipo de heurística, el formato de los datos de entrada y el formato de los datos de salida de la ejecución.

A continuación se muestra la tercera pantalla del proceso de EXECUTE:

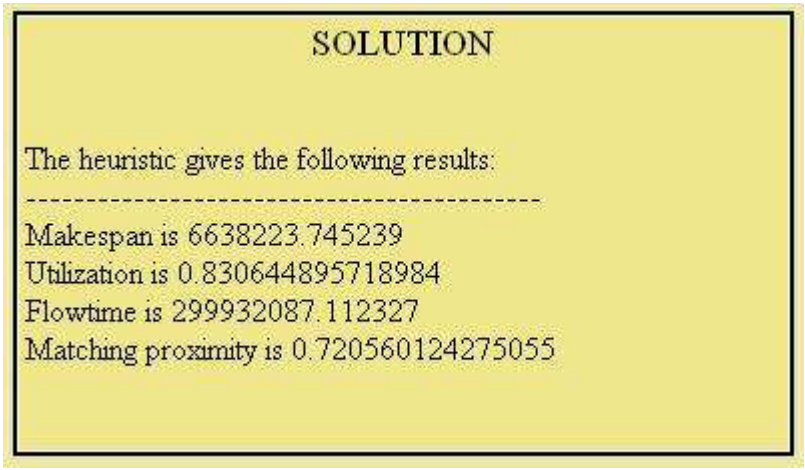


Para finalizar accederemos a la última pantalla donde podremos seleccionar una instancia existente o podremos ejecutar una creada por nosotros mismos. En ambos casos deberemos rellenar un par de campos más: número de máquinas y número de tareas.



Por último pulsaremos sobre EXECUTE y nuestra instancia será tratada por la heurística seleccionada. Dependiendo del formato en el cual hayamos seleccionado que se nos muestren los datos obtendremos un e-mail con los resultados o veremos la solución parcial por la pantalla.

A continuación se muestra la solución parcial obtenida después de realizar una ejecución:

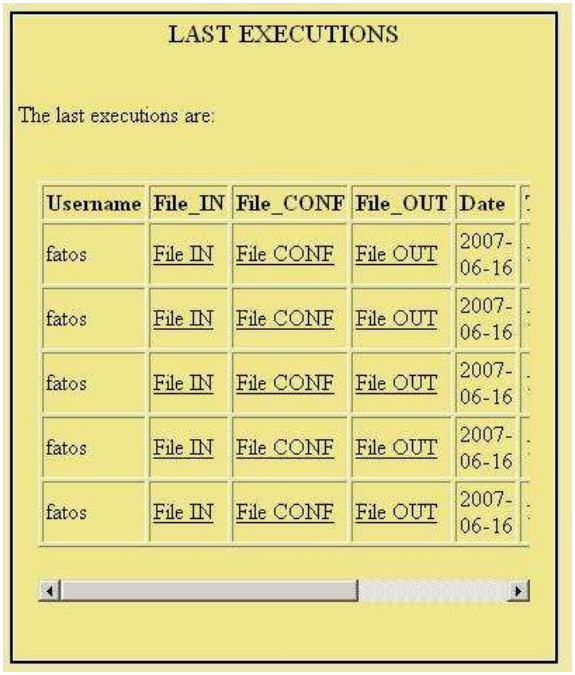


4.5.2.4. I want to see my last executions

Fichero: lastExec.php

En esta sección los usuarios podrán ver cuales han sido las últimas ejecuciones que han realizado. Se mostrará en un listado el nombre de las instancias ejecutadas ordenadas por fecha de ejecución.

A continuación se muestra una figura con las últimas cinco ejecuciones de un usuario:



En el cuadro mostrado podremos encontrar información sobre el usuario que ha realizado la ejecución, podremos ver la instancia que hemos introducido, así como también la configuración que hemos realizado y los resultados que hemos obtenidos ,se muestra la solución completa aunque el usuario no haya seleccionado la opción de recibir la respuesta en su cuenta de correo.

Además el usuario podrá ver datos sobre la ejecución como: la fecha en que la realizó, la hora exacta de la misma, el tipo de heurística seleccionada y si fuese el caso el modo de ejecución.

4.5.2.5. My Account

Fichero: MyAccount.php

En esta sección el usuario podrá gestionar parte de su cuenta de usuario. En una primera pantalla se muestra la información sobre el usuario y se da la posibilidad de cambiar alguno de los datos introducidos en el registro:

Si opta por cambiar el password, la web le llevará a una pantalla donde deberá escribir el password actual de usuario y el nuevo password obligándole a repetir el nuevo password para evitar posibles errores en el mecanografiado del mismo.

Por otro lado si decide cambiar el e-mail no deberá introducir ningún dato identificador, simplemente introducir el nuevo mail y éste substituirá al antiguo.

La última opción que aparece es dar de baja la cuenta de usuario, si selecciona ésta aparecerá una pantalla donde se explica que sin una cuenta de usuario registrado el usuario no podrá acceder a la parte privada de la página. Por último pide la confirmación antes de dar de baja la cuenta de usuario.

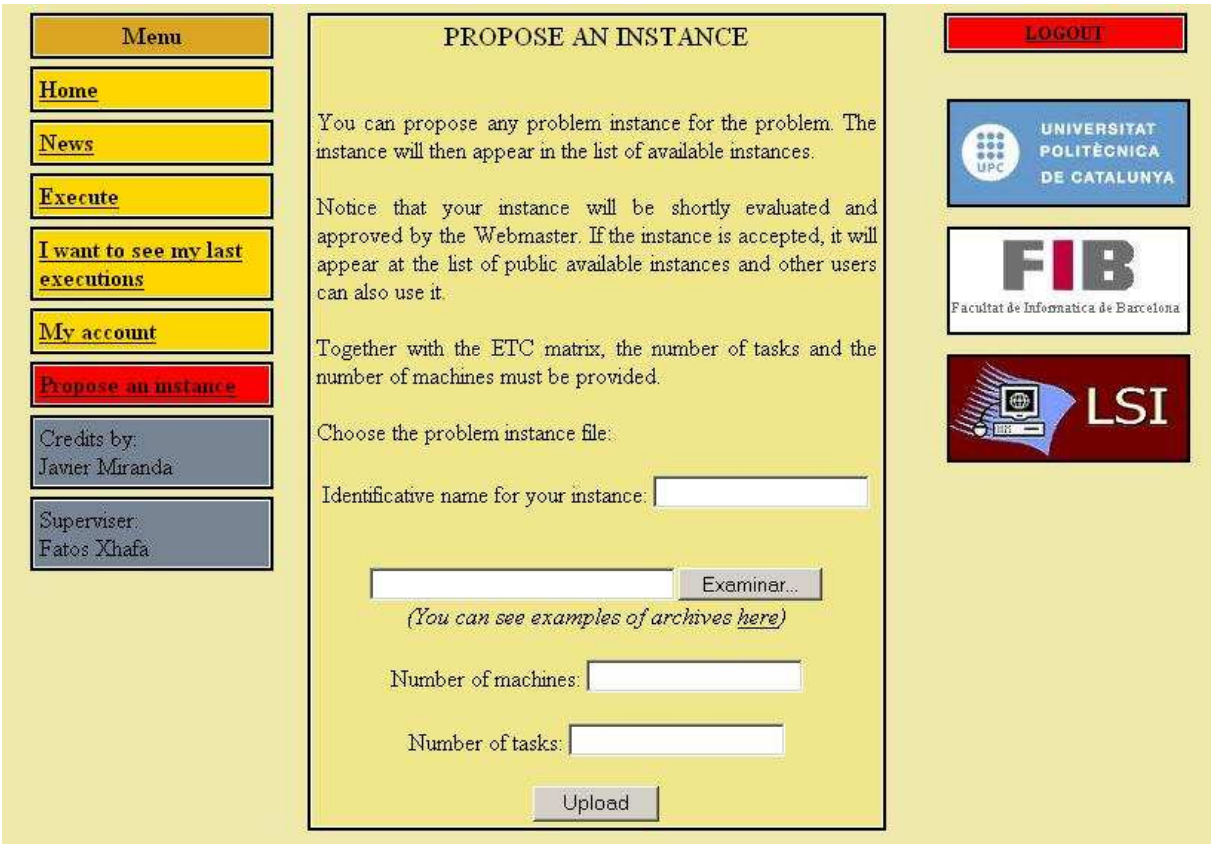
4.5.2.6. Propose an instance

Fichero: proposal.php

Esta última sección solo se encuentra disponible en las cuentas de investigadores y permite proponer una instancia para hacerla pública. La instancia propuesta será mostrada en el apartado de propuestas del administrador para que éste evalué si es una instancia válida o no.

Para poder proponer una instancia el investigador deberá rellenar el formulario que se muestra en la parte central de la página. Deberá escribir un nombre identificativo para la instancia e indicar el nombre de máquinas y tareas por la que esta formada. Por último deberá seleccionar la instancia de sus archivos locales y subirlo al servidor mediante el botón Upload.

A continuación se muestra la pantalla de upload de las instancias:



Por último me gustaría destacar que dentro del formulario se da la opción de mostrar la página donde se explica el formato de las instancias para que éstas sean aptas para la ejecución. Si la instancia es aceptada por el administrador, el usuario verá su instancia junto con el resto de instancias en el momento de seleccionar una instancia de las propuestas por la aplicación.

4.6. Estructura Interna

Toda la estructura que se ha visto hasta el momento es visible para los usuarios, pero la página web necesita de otros ficheros para su correcto funcionamiento. Estos ficheros tendrán por misión ocuparse de tareas más concretas, como puede ser el control de las ejecuciones, el envío de e-mails a los usuarios o también la realización del control sobre las opciones de configuración que el usuario selecciona.

Todos estos archivos los podemos encontrar en el directorio **lib** de nuestro directorio raíz. A continuación se describe la funcionalidad que ofrecerá cada uno de los ficheros por separado y se comentará alguna de sus funciones más representativas.

4.6.1. accesoDenegado.php

Este módulo se encarga de gestionar los errores de acceso a la parte privada de la página. Si algún usuario intenta acceder sin identificación o sin permiso a una parte que no debe de la página web este módulo se encargará de mostrar un error mediante su operación doInfoAccesoDenegado().

La operación mostrará un texto en el cual se indica que para acceder a esta parte de la web hay que disponer de una cuenta de usuario y además dará la opción de acceder a la página de nuevo usuario.

4.6.2. administrador.php

Este módulo es uno de los que contiene más operaciones de todos los módulos del directorio lib, ya que en el se encuentran la gran mayoría de las operaciones que puede realizar el administrador de la web.

Entre las funciones podemos encontrar funciones de gestión de usuarios como doMainDelete() que es la encargada de eliminar usuarios del registro o funciones relacionadas con la lectura, escritura y borrado de noticias.

4.6.3. cabecera.php

Este módulo es el encargado de realizar la cabecera de la página web. La cabecera varía dependiendo del apartado de la página donde nos encontremos.

Con la cabecera conseguimos entre otras cosas saber de forma clara a que partes de la página hemos accedido si miramos en el historial de nuestro navegador.

4.6.4. comprobarmail.php

Este módulo es el encargado de mirar si el mail que nos han introducido en el registro es un mail válido o no.

Para saber si es un mail válido la operación comprobar_Mail() realiza un seguido de comprobaciones entre las que destaco las siguientes:

- Comprobar que la dirección contenga una @.
- Comprobar que la @ no se encuentre ni al principio ni al final del mail.
- Comprobar que la dirección de correo contiene algún punto.
- Comprobar que la dirección no contenga caracteres no válidos.

4.6.5. credits.php

Este fichero se encarga de mostrar los créditos en la interfaz web. Se encarga de realizarlo llamando a su función doCredits(). Su contenido es estático y solo hay que cambiar el código HTML para modificar su contenido.

4.6.6. derecha.php

Este módulo se encarga de realizar la parte derecha de la página web, como hemos comentado con anterioridad este menú es dinámico y a medida que vamos navegando por la web su contenido va cambiando.

Las operaciones más destacadas son:

- doLogin(), en la cual dependiendo del valor que le pasemos cargará el menú de un usuario logueado o de un usuario sin loguear.
- DoRight(), es el encargado de cargar todo el contenido del menú y de indicar a la operación anterior si debe cargar el botón de login de una manera o de otra.

4.6.7. general.php

Este modulo es uno de los que más peso tiene dentro de la librería, es el encargado de cargar la mayoría de datos en la parte pública de la web.

Las funciones que contiene son las siguientes:

- doMainInfo(), contiene los textos mostrados en el Home Page.
- doSchedInfo(), contiene información sobre los problemas de scheduling.
- doParameterInfo(), contiene información sobre los parámetros de entrada en los ejecutables de la web.
- doFileExampleInfo(), informa sobre las instancias que aceptan los programas.
- doAnswerInfo(), informa sobre los resultados obtenidos por las ejecuciones

4.6.8. images.php

Este modulo es el encargado de cargar parte del menú de la derecha. Concretamente se encarga de cargar los links con imágenes.

Si el administrador de la página quisiese añadir algún nuevo link con imagen solo tendría que añadirlo en el código HTML de este módulo y automáticamente se cargaría en todas las secciones de la página web.

4.6.9. infolink.php

Este modulo es el encargado de cargar los datos en el apartado “Interesting Links” de la parte pública de la web. El contenido del apartado lo carga mediante su única función llamada doInfoLink().

4.6.10. infoLoginPage.php

El modulo es el encargado de cargar los datos en la sección de Login a la cual se puede acceder desde todos los puntos de la página web. El contenido al igual que en otros casos es cargado por una única función llamada doInfoLoginPage().

4.6.11. investigador.php

En un principio este modulo debería cargar la mayor parte de las funciones de un investigador al igual que ha pasado con el modulo de administrador, pero en este caso solo contiene la operación que da relleno al Home de investigador. Esto es debido a que durante la implementación se ha decidido crear un módulo que explicaré más adelante llamado opComunes, donde podremos encontrar todas las operaciones que tienen en común el investigador y el invitado.

4.6.12. invitado.php

Nos encontramos ante el mismo caso que antes, inicialmente en este archivo se iban a encontrar todas las operaciones relacionadas con el invitado, pero debido a un cambio durante la implementación para evitar duplicación de código se ha decidido eliminar del modulo las operaciones que tiene en común con un investigador.

Así en este módulo solo encontraremos la función que da la bienvenida a los invitados logueados en la parte privada de la página web.

4.6.13. izquierda.php

Este módulo se encarga de realizar la parte izquierda de la página web, como hemos comentado con anterioridad este menú es dinámico y a medida que vamos navegando por la web su contenido va cambiando.

Por otro lado me gustaría recordar que podemos encontrar cuatro versiones distintas de este menú, de las cuales tres (las de la parte privada) son cargadas en la interfaz web con una misma función llamada doLeftLogin().

Otras operaciones destacadas son:

- doLeft(), se encarga de cargar el menú izquierdo en la página principal, es decir, en la parte pública de la web.
- doLeftCell(), es la función encargada de crear los enlaces en cada botón del menú, además de ser la encargada de decidir si un botón a de mostrarse de un color u otro en el menú.

4.6.14. opComunes.php

Como se ha comentado con anterioridad este módulo ha surgido durante la etapa de implementación y en su interior contiene todas las operaciones comunes de los investigadores y los invitados.

Dentro del módulo podremos diferenciar entre tres tipos de operaciones:

- Operaciones de gestión de MyAccount: Estas son las encargadas de gestionar todo lo relacionado con el usuario: cambio de contraseña, cambio de mail,etc...
- Operaciones de consulta: En este caso encontramos las operaciones de lectura de noticias, o incluso las de consultar las últimas ejecuciones realizadas por el usuario.
- Operaciones de ejecución: En este caso agrupamos todas las operaciones relacionadas con la ejecución de programas. Éste tipo de operaciones son las que más abundan dentro del módulo ya que podemos encontrar diferentes modos de ejecución y para cada uno de ellos se ha debido de implementar una función.

4.6.15. proposal_user.php

Este módulo esta únicamente creado para tratar las propuestas de las instancias realizadas por los investigadores. Dentro del módulo podemos encontrar desde la operación de subir el fichero al servidor hasta la operación que muestra al administrador las propuestas realizadas por los usuarios.

4.6.16. register_info.php

Este módulo es el encargado de mostrar toda la información relacionada con el registro de los usuarios. También es el encargado de gestionar todos los errores que se puedan producir durante un registro como: passwords inválidos, usuario ya existente en la parte privada de la web, etc...

Las funciones que encontramos dentro de este módulo son las siguientes:

- doRegisterInfo(), es la función encargada de mostrar el formulario a los futuros usuarios de la web.
- doErrorRegister(), es la función encargada de avisar a los usuarios que el username utilizado para el registro ya esta siendo utilizado por otro usuario registrado de la página.
- doFaltanCampos(), esta función se encarga de avisar al usuario que ha olvidado rellenar alguno de los campos obligatorios para el registro.
- doErrorPass(), la función nos muestra un mensaje de error donde se nos explica que el password introducido y al repetición de password no coinciden.

4.6.17. remember.php

Este módulo es el encargado de recordarnos la contraseña en caso de que nos olvidemos de ella. Para ello implementa una función de consulta en la base de datos y otra de envío de mail a la dirección del usuario que ha pedido el recordatorio de contraseña.

La función encargada de realizar el envío del mail se llama doSendPass(). Esta operación es la que debemos modificar si deseamos cambiar el mensaje que se envía junto con el recordatorio de contraseña.

4.6.18. screen.php

En este módulo encontramos las funciones encargadas de gestionar algunos datos que son mostrados por pantalla, como por ejemplo los títulos de las secciones.

4.6.19 try.php

Este módulo es el encargado de gestionar la demo que se ofrece en la parte pública de la web. Es el encargado de proporcionarnos el formulario inicial, la matriz ETC para que sea rellenada y de tratar el resultado obtenido para ser mostrado por pantalla.

Las funciones que permiten que permiten que estas funcionalidades se lleven a cabo son las siguientes: doTryInfo(), doSolucion() y llenaMatriz()

4.7. Diseño e implementación de la Base de Datos

Por lo que referencia a la base de datos, como hemos comentado ha sido implementada en MySQL. Esto nos brinda la posibilidad de aprovechar alguna de las opciones que nos aporta una base de datos relacional como la utilizada en este proyecto. Los componentes que hemos utilizado de la base de datos son los siguientes:

Relaciones o Tablas: Nos han permitido guardar la información de la aplicación físicamente. El esquema de la tabla se realiza de la siguiente manera:

Nombre de la tabla (Atributo1, Atributo 2,..., Atributo n, Restricciones)

Un ejemplo sería la tabla utilizada para guardar las noticias publicadas por el administrador:

CREATE TABLE news (name CHAR(30) NOT NULL, notice TEXT, fecha DATE NOT NULL, PRIMARY KEY(name,fecha));

Restricciones: Son las encargadas de establecer condiciones en la base de datos que nos ayuda a controlas los errores por ejemplo de dominio. En el ejemplo anterior se han utilizado dos restricciones: La primera nos dice que la fecha de publicación de una noticia no puede ser nula y la segunda nos dice que el nombre de la noticia es *primary key* o lo que es lo mismo que tiene que ser único ya que la noticia se va a identificar por ese nombre dentro de la base de datos.

Por otro lado hay que comentar la utilización de **triggers** o disparadores que son los encargados de realizar una acción cuando sucede algún acontecimiento. Se dice que este tipo de características de la BD son reglas ACA (Acontecimiento, Condición, Acción) que permiten ejecutar una Acción cuando se produce un Acontecimiento y se cumple la Condición.

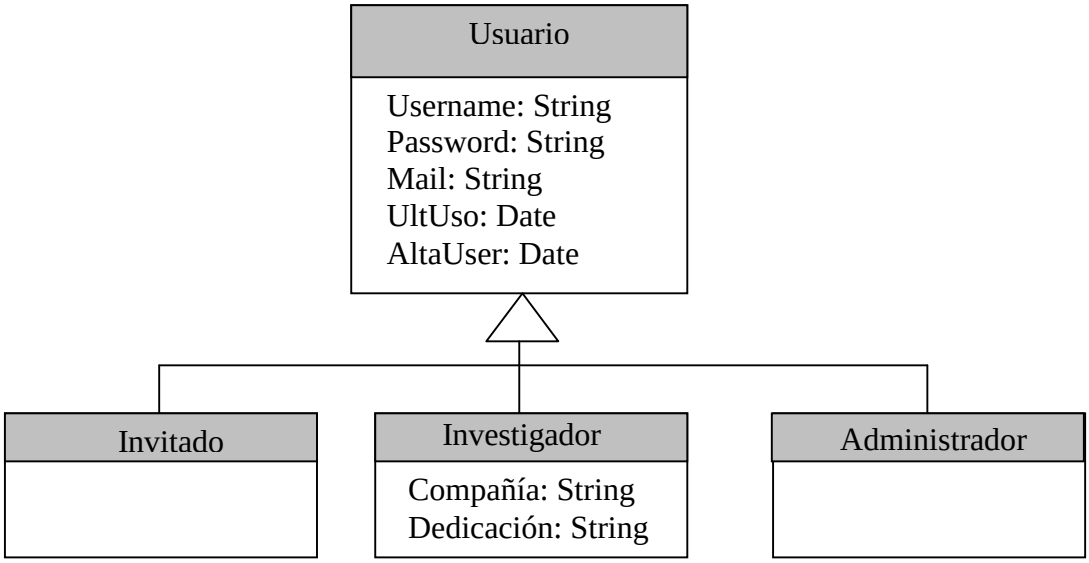
Los triggers han sido utilizados en el caso de las estadísticas, por ejemplo cuando se da un usuario de alta la BD automáticamente se encarga de actualizar la tabla de estadísticas con un nuevo usuario del tipo que se haya creado.

Una característica muy importante de una base de datos relacional es que permite realizar procedimientos guardados (stored procedures). Estos procedimientos son una especie de funciones que son guardados en la propia base de datos. En nuestro caso no ha sido necesario realizar ninguno de ellos, ya que si hubiésemos decidido realizar uno hubiese perjudicado más al proyecto que mejorarlo.

4.7.1. Decisiones sobre la BD

Uno de los primeros problemas que nos encontramos en el diseño de la base de datos, además de la decisión de las tablas que se van a crear, es como van a tratarse las jerarquías de especialización que encontremos en el proyecto.

En nuestro caso solo hemos tenido que realizar una jerarquía, la cual se muestra a continuación:



Las opciones que barajábamos a la hora de crear la jerarquía en la BD eran tres. Podíamos utilizar el método Class Table Inheritance, el Concrete Table Inheritance o el Single Table Inheritance. A continuación explico brevemente en que consiste cada uno de ellos y el motivo por el cual al final decidimos utilizar el Single Table Inheritance.

Class Table Inheritance: consiste en crear una tabla para cada clase del diagrama, así en este caso hubiésemos tenido que crear cuatro tablas en nuestra BD. Descartamos esta opción debido a que la clase Invitado y Administrador no aportaban nada nuevo a la súper clase Usuario, de esta forma complicaciones a la hora de introducir la información en la BD.

Concrete Table Inheritance: consiste en crear una tabla para cada una de las clases concretas de la jerarquía, es decir para Invitado, Investigador y Administrador. En este caso Usuario se quedaría sin tabla en la BD. Todos tendrían que contener la información de los atributos de Usuario dentro de sus tablas. Esta fue una de las opciones que más puntos tunía para ser utilizada, pero como veremos a continuación la última opción es más fácil de gestionar.

Single Table Inheritance: consiste en crear una única tabla, la cual contiene la información de los atributos de Usuario y además contiene los atributos del resto de clases de la jerarquía en este caso los atributos añadidos serían *Compañía* y *Dedicación.* A la tabla se le añade un último atributo llamado *Tipo,* el cual contiene información sobre el tipo de usuario que estamos tratando (Invitado, Investigador o Administrador). Este tipo de tabla nos aporta ventajas ya que es muy simple de utilizar y permite realizar cambios de forma más sencilla. En contra hay que decir que es menos eficiente que las otras dos opciones explicadas. Por último me gustaría destacar que los campos procedentes de las clases concretas, pueden tomar valor nulo, ya que no son utilizados por todos los usuarios.

4.7.2. Tablas de la BD

En la base de datos que se ha diseñado para el proyecto final de carrera se han creado cinco tablas, las cuales podremos ver a continuación junto a una pequeña descripción que nos indicarán porque han sido creadas.

Junto a cada explicación se muestra una figura, en la que la primera fila corresponde al nombre de la tabla, la segunda al nombre de sus atributos y la tercera el tipo de dato con el que se ha creado sus atributos en la base de datos MySQL.

Todos: Es la encargada de guardar información sobre los usuarios registrados en nuestra aplicación. Es el resultado de aplicar el single table inheritance a la jerarquía explicada en el punto anterior de la memoria. El resultado es el siguiente:

todos							
name	cont	tipo	mail	dedica	compan	ultuso	altauser
char(30)	char(30)	char(15)	char(50)	char(50)	char(50)	date	date

El campo name, que es la clave primaria de la tabla, contiene el username, el campo comt contiene el password del usuario, el campo tipo indica el tipo de registro que realizó el usuario, el campo mail contiene la dirección de correo del usuario, dedica contiene la dedicación y compan la compañía para la cual trabaja/estudia.

Los dos últimos campos de la tabla contiene información sobre la ultima vez que el usuario ha accedido a la aplicación y sobre cuando realizó el alta de usuario.

News: Es la tabla encargada de guardar las noticias publicadas por el administrador, su campo *name* es el campo identificador de la noticia y junto a su campo *fecha* son los dos campos que nunca pueden ser nulos ya que ambos forman la clave primaría de la tabla. Mediante el campo fecha además conseguimos controlar el tiempo que pasa una noticia publicada en la aplicación. Por último comentar que el campo *noticia* es el que contiene la noticia que será publicada en la sección news de la aplicación. El resultado es el siguiente:

news		
Name	noticia	fecha
char(30)	text	date

Accesos: Esta tabla es la encargada de gestionar las entradas de los usuarios a la parte privada de la aplicación. Cada vez que un usuario accede a la página se crea una instancia de la tabla. Los accesos son borrados cada tres horas o cada vez que un usuario se desloguea de la parte privada. De esta forma se evita que un usuario mal intencionado acceda a la parte privada de la página sin tener autorización. Los campos que podemos encontrar en la tabla son *user (clave primaria)*, que nos indica el usuario que ha realizado el acceso. *Timelog* y *datelog* que nos indican respectivamente a que hora y que día se ha realizado el acceso a la parte privada. El resultado es el siguiente:

accesos		
user	timelog	datelog
char(30)	time	date

Ejecuciones: Es la encargada de registrar las ejecuciones realizadas en la aplicación. Solo guarda las ejecuciones que se realizan en la parte privada de la página, es decir las ejecuciones realizadas en la demo de la parte pública no se registran. Sus atributos son: *user* que contiene el nombre del usuario que ha realizado la ejecución. El campo *nomdir* contiene información del directorio donde se ha guardado tanto el fichero de entrada, el de configuración y el de salida de la ejecución realizada. Los campos *fecha* y *hora* contienen respectivamente la fecha y la hora en que se ha ejecutado la aplicación. El campo *tipoheur* contiene información sobre el tipo de heurística ejecutada y por último, el campo *modo* contiene información sobre el modo de heurística en caso de que la heurística fuera de tipo Ad-hoc. El resultado obtenido es el siguiente:

ejecuciones					
user	nomdir	fecha	hora	tipoheur	modo
char(30)	char(50)	date	time	char(30)	char(30)

Debemos destacar que esta tabla no tiene clave primaria, así que el propio gestor de la base de datos se encarga de proporcionar una. La clave que habitualmente proporciona el SGBD suele ser un numero identificativo al principio de la tabla.

Por otro lado el campo nomdir contiene, como se ha comentado, información sobre el directorio donde se ha guardado la ejecución, accediendo al contenido de este campo podemos obtener todos los datos de la ejecución: fichero de entrada, configuración y resultados.

Estadísticas: Es la última tabla creada en el proyecto, su función es realizar el control de las estadísticas. En ella guardamos el número de ejecuciones que se han realizado en la aplicación desde su entrada en funcionamiento, el número de ejecuciones desde una fecha establecida por el administrador, la fecha establecida por el administrador para el control de estas ultimas ejecuciones, el número de usuarios de todos los tipos que se encuentran registrados actualmente en la aplicación y por último el número de usuarios borrados en la aplicación.

estadísticas			
numejec	numejecpar	dateejecpar	numbor
integer	integer	date	integer

Hay que destacar que esta última tabla es controlada casi en su totalidad por el lanzamiento de triggers los cuales se activan cuando sucede algún evento en alguna de las otras tablas que forman la base de datos.

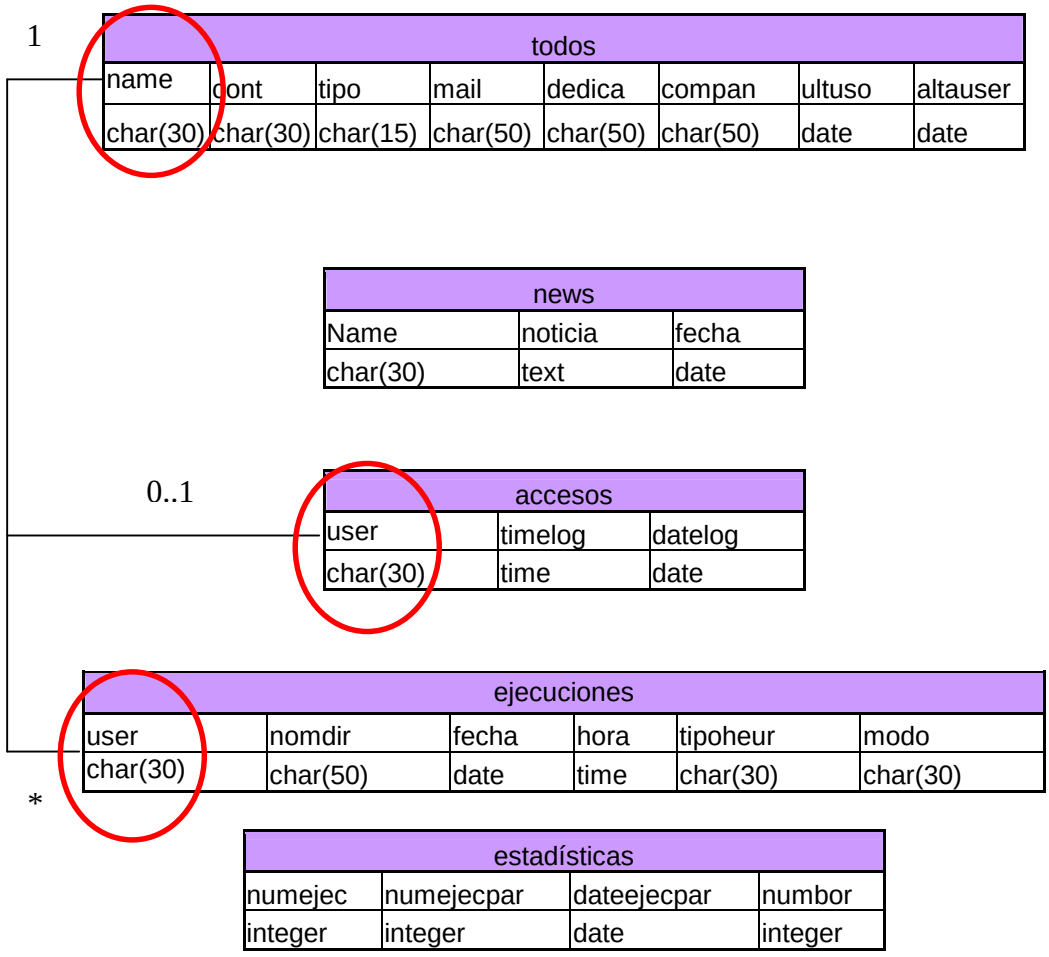
También hay que destacar que esta tabla tampoco dispone de clave primaria.

Por último y para acabar con este apartado me gustaría comentar que se pensó en realizar una sexta tabla en la base de datos. Esta tabla debería contener la información introducida por el administrador sobre el tipo de ejecución que deberían realizar los invitados/investigadores y con las características de cada uno de los usuarios a la hora de ejecutar las aplicaciones ofrecidas, es decir, la tabla controlaría el apartado “Configuration User” del administrador.

Esta tabla fue substituida por un fichero llamado **formUser.xml**, el cual podemos encontrar en el directorio raíz de la aplicación. Cada una de las líneas corresponde a un campo de la configuración de los usuarios. Las líneas del archivo tienen estas funciones:

1. Contiene el tipo de ejecución que se realizará, puede ser secuencial o paralela.
2. Contiene el número máximo de ejecuciones de un invitado.
3. Contiene el tiempo máximo de ejecución de un invitado.
4. Contiene el número máximo de ejecuciones de un investigador.
5. Contiene el tiempo máximo de ejecución de un investigador.
6. Contiene el número de ejecuciones realizadas que puede ver un usuario en el apartado “I want to see my last executions”.

4.7.2.1 Esquema de relaciones:



El campo name de la tabla todos es referenciado por el campo user tanto de la tabla ejecuciones como de la tabla accesos. Las tablas news y estadísticas no mantienen ninguna relación con el resto de tablas.

4.7.3 Ficheros

Como en el caso del fichero formUser.xml, a lo largo de la realización del proyecto nos hemos visto obligados a realizar algún archivo más para el control de la aplicación. Los archivos que hemos tenido que crear han sido tres los cuales explicaremos a continuación:

4.7.3.1. accesoBD.xml

Es el primero de los archivos y el más importante, ya que de su correcta configuración depende el buen funcionamiento de toda la aplicación. Este archivo contiene información sobre la base de datos a la cual debemos acceder para encontrar todas las tablas explicadas hasta el momento.

El archivo esta creado de la misma forma que el formUser.xml, cada línea del archivo corresponde a una información. A continuación se explica el contenido de cada una de las líneas.

1. Máquina donde se encuentra alojada la base de datos, en nuestro caso *davinci.lsi.upc.edu*, aunque si fuese una aplicación a nivel local podríamos poner en esta línea *localhost* por ejemplo.

2. El nombre del usuario de la Base de datos, en nuestro caso jmiranda.

3. El password para acceder a la Base de datos correspondiente al usuario de la línea anterior.

4. El nombre de la BD a la cual tenemos que conectarnos, en este caso la BD coincide con el nombre de usuario: jmiranda.

El motivo por el cual se creó este archivo es para facilitar el cambio y la portabilidad de la aplicación. Si decidiésemos instalar la aplicación en cualquier otro lugar o simplemente cambiar la base de datos a otra máquina, solo deberemos modificar este archivo y todos los accesos que se realicen a la base de datos se realizará con la información contenida en el archivo.

Por último mencionar que es necesario que el archivo accesoBD.xml se encuentre en el directorio raíz de la aplicación, sino la aplicación no podría interpretarlo y por lo tanto no tendríamos acceso a la base de datos.

4.7.3.2. crear_my.php

Este fichero es el encargado de crear la base de datos que trataremos desde la aplicación. Es conveniente ejecutar este archivo una única vez, aunque si lo ejecutamos más de una vez no pasaría absolutamente nada, ya que nos indicaría que no se han podido crear las tablas debido a que ya existen.

La estructura del fichero en este caso es diferente a los xml explicados hasta el momento, en este caso nos encontramos ante un fichero escrito en PHP. El resultado de acceder a esta página mediante el navegador es que se crearan las siguientes tablas: todos, news, accesos, ejecuciones y estadísticas (explicadas en el apartado 4.7.2 de la memoria) en la base de datos configurada en el archivo accesoBD.xml.

El archivo también se encarga de crear un usuario administrador, para poder controlar toda la aplicación inmediatamente después de ejecutar este fichero. El usuario creado es “jmiranda” y con contraseña “password”.

Hay que destacar que la única manera de crear usuarios administradores es mediante un fichero auxiliar explicado en el manual del administrador (Anexos), ya que desde la aplicación web no se permite la creación de este tipo de usuario.

Por último para acabar el comentario sobre este fichero deberiamos destacar que una vez realizada la ejecución del fichero es recomendable eliminarlo del espacio público, ya que contiene información muy golosa para algunos utilitarios de la red (hackers).

4.7.3.3. borrar_tablas_my.php

Al igual que el fichero anterior este fichero se encuentra escrito en PHP, pero podríamos decir que nos encontramos ante el antónimo del fichero anterior. Si el otro fichero estaba dedicado a la creación de la base de datos que va a funcionar con nuestro proyecto, este fichero se dedica a todo lo contrario: a la destrucción de la base de datos.

El resultado de ejecutar el fichero borrar_tabalas_my.php es que borraremos de la base de datos todas las tablas de la aplicación incluido su contenido, es decir, eliminaremos todos los usuarios, incluido el administrador, y todos los datos que se han ido recopilando de cada uno de ellos.

Al igual que en el caso anterior para ejecutar este archivo debemos acceder mediante el navegador y una vez ejecutado sus cambios son irrecuperables.

Es recomendable no mantener este archivo en el espacio publico, ya que es altamente peligroso y un acceso no deseado a él puede dejar inservible la aplicación. También es recomendable que este archivo se mantenga fuera del servidor y solo subirlo al servidor en el momento que se deseé destruir todos los datos de la aplicación (en caso de que esto se quiera realizar alguna vez).

Una vez se accede a la página borrar_tablas_my.php el navegador nos mostrará un seguido de mensajes indicándonos si se ha podido borrar o no las tablas que forman la base de datos.

4.8. Otros aspectos de diseño

La aplicación que se ha realizado a lo largo de proyecto no ha sido diseñada para implementarse en un único sistema, en este caso la aplicación ha sido diseñada para un sistema distribuido y he considerado apropiado antes de acabar el diseño de la aplicación explicar en que consisten los sistemas distribuidos.

Un sistema distribuido básicamente es una colección de nodos autónomos conectados mediante una red física. Cada nodo ejecuta componentes y usan un *middleware* que permite conectar y coordinar todas las actividades de los componentes de manera que los usuarios ven el sistema como un único recurso informático integrado.

4.8.1. Características de los sistemas distribuidos:

Como todo tipo de sistema los sistemas distribuidos también aportan ventajas y desventajas respecto a otros tipos de sistema, a continuación se muestran las características más destacadas de estos sistemas:

Ventajas:

- Permite la compartición de recursos.
- Permiten ejecuciones concurrentes
- Son altamente escalables.
- Son muy tolerantes a fallos

Desventajas:

- Son muy complejos
- Disminuyen la seguridad de la aplicación
- Producen un mayor esfuerzo de gestión
- Pueden ser impredecibles.

Un middleware es una colección de componentes de infraestructura que permiten a los componentes del dominio de un sistema comunicarse los unos con los otros a través de la red. Los middlewares habitualmente proporcionan dos tipos de servicios: de desarrollo y de administración.

El middleware que tratamos en el proyecto es de tipo MOM y tiene las siguientes características:

- El cliente pública un mensaje en una cola de mensajes.
- El servidor recoge los mensajes y reacciona según el tipo de mensaje.
- Tiene una arquitectura más desacoplada que los middlewares RPC.

4.8.2. Fronteras de distribución

Resulta difícil hablar sobre sistemas distribuidos sin comentar nada sobre fronteras de distribución. Se llaman fronteras de distribución a la separación física que existe entre los componentes de la arquitectura de un sistema.

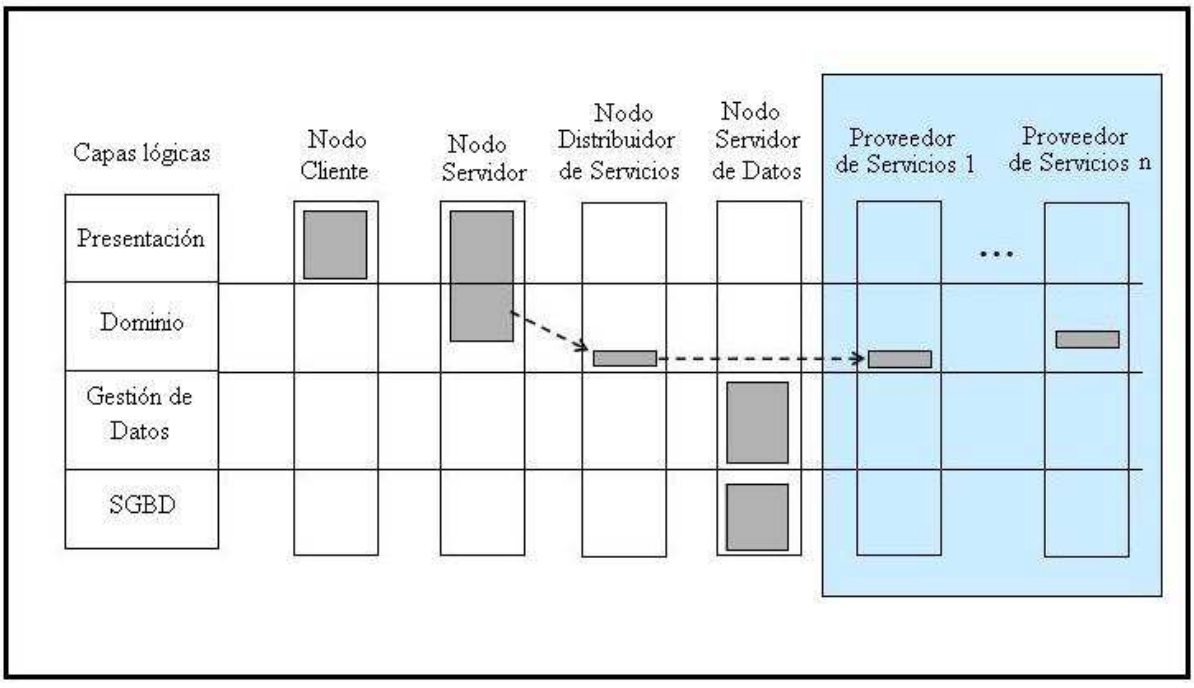
En una arquitectura en tres capas, las fronteras de distribución pueden:

- Asignar capas diferentes a nodos diferentes.
- Asignar capas diferentes a un mismo nodo.
- Distribuir una misma capa entre más de un nodo.

En nuestro caso las fronteras que hemos tratado en el proyecto son las siguientes:

- Presentación distribuida
- Datos remotos
- Distribución de la capa de dominio.

A modo de ejemplo se muestra la siguiente figura:



En nuestra aplicación el esquema anterior se correspondería a lo siguiente:

Nodo cliente → Cliente que accede mediante el navegador.
Nodo servidor → Máquina ba0.lsi.upc.edu del laboratorio de cálculo del LSI.
Nodo distribuidor de servicios → Máquina sobre la que se ejecuta CONDOR
Proveedores de servicios → Todas las máquinas a las que accede el condor.
Nodo servidor de Datos → Máquina davinci.lsi.upc.edu donde se encuentra la BD del proyecto.

Por último y para cerrar el capítulo de diseño comentaremos brevemente alguno de los patrones de diseño que se han aplicado a la capa de presentación de la aplicación. Básicamente se han intentado seguir dos patrones simultáneamente, el primero el *patrón protección* que es más orientado a evitar errores por parte del usuario. El segundo el *patrón guía* que busca facilitar en medida de lo posible la facilidad de los casos de uso del usuario.

La idea de aplicar el patrón protección surge cuando en la aplicación han aparecido ciertas acciones que tienen efectos importantes, o más bien dicho irreversibles, sobre el sistema. O bien, porque alguna acción tenga un coste muy alto si tratamos de deshacerla.

Por ejemplo el administrador podría estar navegando por la aplicación y entrar por accidente en el apartado dedicado al borrado de usuarios, lo que se intenta con este patrón es que sea poco probable que por el hecho de entrar en ese apartado el administrador borre a un usuario sin desearlo.

Para dar solución a este tipo de problemas y a otros semejantes se ha intentado seguir a lo largo del proyecto estas dos reglas:

- Añadir un nivel extra de protección a la función, es decir, el usuario debe equivocarse dos veces para realizar una acción que tenga efectos importantes.

- El usuario debe confirmar explícitamente la opción elegida, por ejemplo: nunca se mostrará por defecto un usuario en la opción de borrar, el administrador deberá seleccionar al usuario de la lista.

El patrón guía es más sencillo de aplicar, ya que consiste en mantener un cierto orden lógico en las diferentes ventanas o menús que se van tratando cuando un usuario realiza un caso de uso. Este patrón también intenta ayudar al usuario en el caso de introducción de datos ofreciendo, si cabe la oportunidad, las opciones con las que rellenarlo.

5. Aspectos del desarrollo

El paso que pasa de la etapa del diseño a la etapa de la implementación ha comportado un seguido de toma de decisiones y algunos problemas que se comentan en la sección 5.1. En la sección 5.2 se habla sobre la seguridad del sitio web y en la sección la 5.3 se proporcionan algunas porciones de código a modo de ejemplo para entender el funcionamiento de la interfaz web.

Para acabar el capitulo se explicará el despliegue, el testing y la explotación de la aplicación.

5.1. Decisiones y problemas

El principal problema que nos encontramos a lo largo del proyecto es cuando llegamos al momento de realizar las ejecuciones. En este momento nos vemos obligados a realizar un estudio de los parámetros de entrada de los ejecutables y sus valores de retorno.

Por otro lado vemos que para realizar una ejecución paralela, mediante CONDOR, debemos suministrar un fichero con toda la información necesaria para que esta ejecución pueda llevarse acabo. Así se llega a la conclusión de que debemos diferenciar las ejecuciones de los distintos usuarios que tenemos en la aplicación.

La solución que decidimos tomar fue la de crear un directorio llamado **jobs** donde se guardarían todas las ejecuciones realizadas. Las ejecuciones a su vez se guardarían en un directorio con el nombre de su usuario y cada ejecución sería identificada por el momento en que se realizará la ejecución, es decir por la fecha y hora. Esta decisión fue tomada porque así se consiguen identificar de una manera única todas las ejecuciones, además el hecho de identificar además la ejecución por su usuario impide que se puedan confundir dos ejecuciones realizadas en el mismo instante, ya que es imposible que un mismo usuario realice dos ejecuciones simultáneamente.

Hay que destacar que el directorio del usuario se crea en el momento del registro del usuario y podemos asegurar que no encontraremos dos usuarios con el mismo nombre ya que el registro no permite que esto suceda. Los directorios que se encuentran dentro de éste, como hemos dicho serán identificados por la fecha y hora de la ejecución y su nombre será formado a partir de la siguiente cadena:

“año + mes + día + hora + minuto + segundo” del momento en que se realice la ejecución.

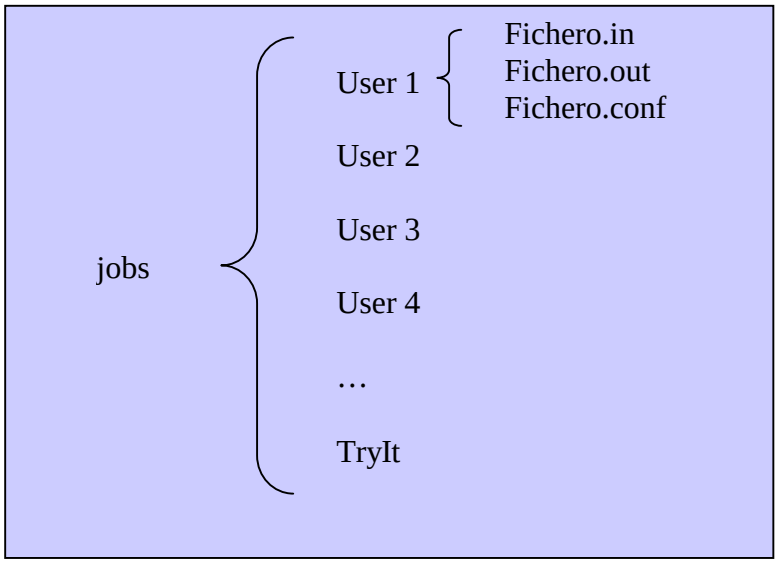
Por otro lado se decidió que estos subdirectorios creados se mantuvieran, aunque no fuesen necesarios una vez el usuario ha recibido la respuesta, por si un usuario perdiese la información del resultado o decidiera recibir la información por mail en vez de por pantalla. En este caso la ejecución estaría bastante bien identificada en el tiempo y se podría enviar un e-mail a un administrador pidiéndole que le facilitará el resultado de nuevo.

Como segunda decisión importante que tuvimos que tomar nos encontramos el ¿Qué mostrar como resultado de una ejecución?, en este caso decidimos que si el usuario deseaba recibir el resultado por la pantalla solo obtendría una solución parcial, en la cual no se le indicaría la asignación de cada tarea a cada máquina. Mientras que si el usuario decidía recibir la información vía e-mail obtendría el resultado completo. La solución parcial la tuvimos que adoptar debido a que algunas ejecuciones producen en resultado muy grande, el cual no sería cómodo de consultar solo en el navegador.

Para finalizar con las decisiones, la última que tuvimos que tomar fue ¿Qué guardar de cada ejecución?, así que decidimos guardar un fichero con el contenido de la entrada al ejecutable (instancia) el cual denominamos *nombrefichero.in*, otro fichero con información sobre los parámetros de entrada llamado *nombrefichero.conf*, y por último un fichero llamado *nombrefichero.out* donde se guarda la salida mostrada por el ejecutable, es decir, la solución.

Cada uno de estos ficheros se guarda en los directorios comentados anteriormente, así que cada directorio de ejecución contendrá tres ficheros en su interior: un in, un out y un conf.

La estructura de directorios de jobs queda de la siguiente manera:



Esta última decisión fue tomada con vistas a que el usuario podría poder volver a consultar sus ejecuciones realizadas, así podríamos facilitarle la instancia que introdujo, la configuración que realizó y el resultado que obtuvo.

Hay que destacar que inicialmente encontramos un usuario creado en el directorio jobs, llamado TryIt, este usuario es dedicado exclusivamente para la demo del programa y es el único caso en que no se guarda el fichero de salida ni el de configuración de las ejecuciones.

Por último comentar que los ejecutables a los cual accede la aplicación nos han producido muchos problemas, ya que se ha tenido que retocar la forma de introducir los datos

dentro del ejecutable para que se pudiera realizar desde la línea de comandos. Los problemas llegaron cuando los ficheros fuentes dieron problemas en la compilación y algunos de los ejecutables después de compilar han dado varios problemas en las ejecuciones.

A parte de estos problemas y decisiones comentados me gustaría mencionar que en el momento de inicialización del proyecto no se tenía conocimiento ninguno sobre el lenguaje PHP y muy poco sobre HTML. Lo cual me ha causado bastantes problemas a la hora de implementar. Por suerte la mayoría de problemas les había surgido a otros usuarios de PHP y habían sido solucionados en foros de Internet.

5.2. Seguridad

Se puede decir que la mayoría de la seguridad de la aplicación web viene dada por la seguridad del PHP y de cómo esté configurado éste. Esto significa que si se detecta un bug en el código del PHP podría utilizarse para atacar el sitio web. Es responsabilidad del administrador del servidor mantenerse informado en todo momento sobre los problemas detectados en las diferentes versiones de los programas utilizados en el servidor, en nuestro caso PHP, Apache y MySQL, y de cómo resolverlos.

La única seguridad que he podido aplicar a la aplicación como diseñador ha sido la parte de lógin o mejor dicho la parte privada de la aplicación, en la cual se ha realizado todo lo posible para que no se puede acceder sin identificación.

Si intentamos acceder directamente a una página de la parte privada de la aplicación desde la barra de direcciones sin identificarnos nos encontraremos con este mensaje:

Por otro lado se han intentado tratar el máximo de errores producidos en la introducción de datos en los formularios, realizando comprobaciones en algunos de ellos.

Como último apartado de la seguridad volver a repetir que los archivos crear_my.php y borrar_tablas_my.php deberían de quedar siempre fuera del alcance de los usuarios de Internet y no ponerlos en el espacio publico si no es necesario.

5.3. Código de Ejemplo

En este apartado se mostrarán y explicarán unas porciones de código extraído del proyecto explicando el funcionamiento del mismo y como se ha integrado el lenguaje PHP en las páginas HTML.

Para empezar me gustaría decir que el PHP funciona como una ampliación de HTML, ya que nos permite disponer del lenguaje HTML pero añadiendo funcionalidades y variables. Hay que destacar que las variables en PHP vienen precedidas por el símbolo del dólar ($) y qué no hace falta realizar ningún tipo de declaración de la variable, ya que el propio lenguaje es el encargado de decidir el tipo de variable que se ha creado, cambiándola de tipo en el momento que fuese necesario.

Para empezar a escribir el código PHP siempre debemos abrir el código con el siguiente código identificativo **<?php** y para cerrar el código PHP deberemos escribir **?>**. Comentar que dentro de una función PHP se pueden abrir y cerrar tantas veces como haga falta el código, de esta forma podemos intercalar el HTML con el PHP incluso dentro de funciones.

5.3.1. Código Base

En la siguiente página podemos observar un código base de la aplicación, la mayoría de las páginas accesibles desde la aplicación mantienen la misma estructura.

En el ejemplo podemos observar el código del apartado de la página “Interesting Links” que contiene los links a otros lugares de interés.

```
<html>
        <?php
        doHtmlHead("Links");
        ?>

<body bgcolor="#EEE8AA" link="#000000" vlink="#000000" onLoad="Hora()">

  <div align="center">
  <table border="0" cellspacing="0" cellpadding="0">
  <tr>
  <?php
  doSuperior();
 ?>
  <tr>
        <?php
        //Crea la parte izquierda de la web
        doLeft("LINKS");
        ?>

        <td valign="top">
        <table border="0" cellspacing="5" cellpadding="2" width="400">

          <?php
                //Llama a la función central
                doInfoLink();
          ?>

        </tr>
        </table>
        </td>

        <td valign="top">
        <table border="0" cellspacing="0" cellpadding="0">

                <?php
                //Inicializa la parte derecha de la página
                doRight("LINKS");
                ?>

        </table>
        </td>
  </tr>
  </table>
  </div>
</body>
</html>
```

En la parte superior del código encontramos la carga del nombre de la página en la cabecera del navegador mediante la función DoHtmlHead(). A continuación como podemos observar la página se dibuja en base a una tabla principal diseñada tal y como se explicó en el apartado 2.3 en esta misma memoria. La tabla se define mediante la etiqueta <table> y puede ir acompañada de algunas características de la tabla como puede ser el cellspacing que se encarga de definir el espacio entre las celas que forman la tabla.

Una vez nos encontramos dentro de la tabla debemos de aprender a movernos a través de ella, para ello utilizaremos las etiquetas <tr> y <td> que nos desplazan por las filas y columnas de la tabla respectivamente.

El siguiente paso es cargar en la primera fila de la tabla la imagen superior de la página, la imagen es la encargada de centrar el flujo de ejecuciones en el centro de la pantalla del usuario. La carga de esta imagen la realizamos con la función doSuperior().

Una vez cargada la imagen empezamos a cargar el contenido de la página, para ello llamamos a la función doLeft() que nos carga el menú de la izquierda, el parámetro que le pasamos le indica en que apartado del menú nos encontramos. Continuando con la carga de datos pasamos a la carga de la parte central de la página, en este caso se llama a la función doInfoLink(), la cual carga la información sobre todos los links que podemos visitar desde nuestra aplicación web.

Finalmente se realiza la carga de la parte derecha de la página, para ello hemos llamado a la función doRight(). Una vez cargado todo el contenido de la página solo nos queda cerrar las etiquetas abiertas como <table>, <body> y por último <html>.

Me gustaría hacer notar que cada vez que se ha llamado a una función hemos introducido el código dentro del PHP, abriendo y cerrando el código PHP mediante las instrucciones explicadas anteriormente.

5.3.2. Control de errores y selección de acciones

A continuación mostramos una porción de código en la cual se realiza el control de errores en el momento en que un usuario sube una propuesta para que esta sea analizada por el administrador.

En este caso podemos observar como PHP trata las variables con el símbolo del dólar delante, la porción de código que veremos no ha intercalado en ningún momento código HTML.

```
<?php
	$error=0;
	$directorio = "/SCHEDGRID/proposals/".$name_inst;

		//Comprobar que el nombre de la propuesta es único
		if(!mkdir($directorio,0777))
			$error=4;

		$fichero=$_FILES["MyFile"]["name"];
		$temporal=$_FILES["MyFile"]["tmp_name"];
		$fichero_ruta=$directorio."/".$fichero;


		//Comprobar que esta subido de forma temporal
		if(!is_uploaded_file($temporal))
				$error=1;

		// Comprobar que no exista
		if (file_exists ($fichero_ruta))
				$error=2;

		//Mover a la ruta correcta
		if (!move_uploaded_file($temporal,$fichero_ruta))
				$error=3;

		if ($error==0){
			doProposalOk();
			}

		else{
			doErrorProposal($error);
			}
 ?>
```

En este código se controla que no suceda ningún error mientras el usuario sube la instancia propuesta al servidor, para ello se realizan una serie de comprobaciones:

1. Comprobamos que el nombre propuesto sea único, ya que identificará la propuesta.

2. Miramos que el fichero se haya subido al servidor de forma temporal

3. Comprobamos que el fichero no se copiado ya al apartado de propuestas.

4. Comprobamos que se mueva correctamente a la ruta deseada desde el espacio temporal del servidor.

Por último si no ha habido ningún error lanzamos un mensaje mediante la función doProposalOk() que informa al usuario que ha subido correctamente su instancia.

En caso de error la función doErrorProposal() se encargará de mostrar al usuario el error correspondiente, ya que cada error de los tratados se identifica mediante un código numérico.

Para acabar con este apartado hay que destacar las facilidades que nos aporta PHP en algunos aspectos de programación, por ejemplo en la concatenación de strings, donde simplemente realizando una asignación de los dos valores a concatenar con un punto entre los dos valores ya quedan concatenados. También mencionar que el trato de condicionales y bucles es exactamente igual que en cualquier otro lenguaje de programación no orientado a la web. Algo que facilita el autoaprendizaje de este lenguaje.

5.4. Despliegue

La aplicación web quedará instalada en el laboratorio de cálculo del departamento del LSI. Concretamente quedará instalada en la máquina ba0.lsi.upc.edu del mismo. Para acceder a la aplicación deberemos hacerlo mediante la siguiente url:

http://weboptserv.lsi.upc.edu/SCHEDGRID/

La función del laboratorio de cálculo es dar soporte a la investigación y la enseñanza tanto de los investigadores como de los profesores, así como también ayudar en los proyectos de los estudiantes que finalizan su carrera.

Los servicios proporcionados por el LCLSI incluyen asistencia técnica a los usuarios, control de los más de 300 equipo que tiene a su disposición, red y seguridad para la misma, gerencia de las cuentas de los usuarios, E-mail, FTP , Web y DNS.

5.5 Testing

Como es habitual en un proyecto, a medida que se ha ido desarrollando el proyecto se han ido realizando pruebas sobre sus funcionalidades. Este apartado está dedicado a las pruebas que se han realizado.

Las pruebas básicamente se han centrado en dos puntos: evitar que los usuarios puedan acceder a la parte privada sin identificación y evitar errores en las ejecuciones debido a la introducción de datos incorrectos en los formularios.

Para el primer punto se ha intentado acceder a todas las secciones de la parte privada sin realizar la identificación previa y se ha comprobado que no es posible.

Para el segundo caso hemos introducido datos erróneos en los formularios y hemos visto como respondía la aplicación, una vez probadas las situaciones hemos solventado los errores que se producían.

En este último punto nos encontrábamos con errores de dos tipos. Los primeros eran errores en la implementación de la aplicación los cuales han sido resueltos, un ejemplo de estos errores es por ejemplo: No mostrar un *Radio Button* seleccionado por defecto, esto

producía que si el usuario no seleccionaba ninguna de las opciones la aplicación diera un error. Para solucionarlo se muestra siempre una opción seleccionada por defecto.

Como segundo tipo de error al que nos enfrentábamos era que el usuario introdujese de forma errónea los datos en los campos de un formulario. Para ello se han aplicado dos soluciones, en primer lugar se muestra un valor siempre en el campo a modo de ejemplo, así el usuario sabrá que tipo de dato debe introducir. Por otro lado, si aún y saber el tipo de dato que debe introducir el usuario introduce un dato erróneo la aplicación puede actuar de dos formas: mostrando un mensaje de error al usuario o omitiendo el dato introducido. La segunda opción solo se aplica en casos donde el dato no es necesario obligatoriamente, o se puede establecer un valor por defecto.

5.6. Explotación

En este apartado se explicará los datos que quedan inicialmente cargados en la aplicación.

Inicialmente se crea un usuario administrador llamado *jmiranda*, además se crea una instancia en la tabla estadísticas la cual se va modificando a medida que van avanzando las ejecuciones y van aumentando los usuarios registrados.

Las heurísticas disponibles para los usuarios son las Tabú Search y las Heurísticas Adhoc, tanto en immediate mode como en batch mode. La aplicación inicialmente muestra unas instancias públicas, las cuales fueron realizadas para ejecuciones de 512 tareas en 16 máquinas (el máximo permitido), de esta forma aseguramos que cualquiera de esas instancias es válida para cualquier ejecución de N máquinas y M tareas.

En la carpeta **jobs** encontramos inicialmente creado un usuario *TryIt*, el cual guarda información sobre las demos ejecutadas en la aplicación.

Por último los ficheros de configuración **accesoBD.xml** y **formUser.xml** se encuentran configurados correctamente para el funcionamiento de la aplicación. El fichero accesoBD.xml tiene configurada la máquina davinci.lsi.upc.edu para acceder a la base de datos. Mientras que formUser.xml tiene como configuración establecida la siguiente:

Número de ejecuciones de invitado: 90
Número de ejecuciones de investigador: 1000
Tiempo invitado: 50
Tiempo investigador: 400
Últimas ejecuciones mostradas: 5

6. Análisis de costes económicos y planificación

En todo proyecto se debe realizar una estimación de tiempo para controlar el trabajo realizado y el que queda por realizar. Este proyecto no ha sido menos que el resto y como tal ha incorporado también una planificación, la cual se ha intentado seguir en medida de lo posible.

En este apartado se ha intentado realizar una comparación entre la planificación inicial que se hizo del proyecto y la planificación final que ha surgido. Por otro lado también se ha intentado calcular el coste del proyecto si se hubiese realizado para una empresa particular en lugar de cómo proyecto final de carrera.

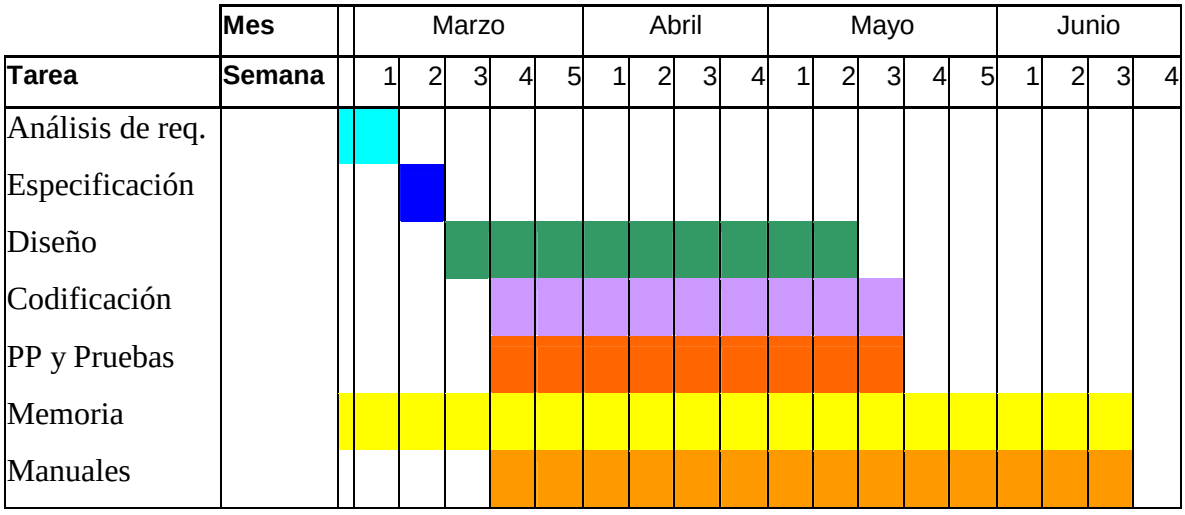
6.1. Planificación

El desarrollo del proyecto se ha realizado dentro del intervalo previsto (un cuatrimestre), aunque con un poco de retraso respecto a la planificación inicial realizada.

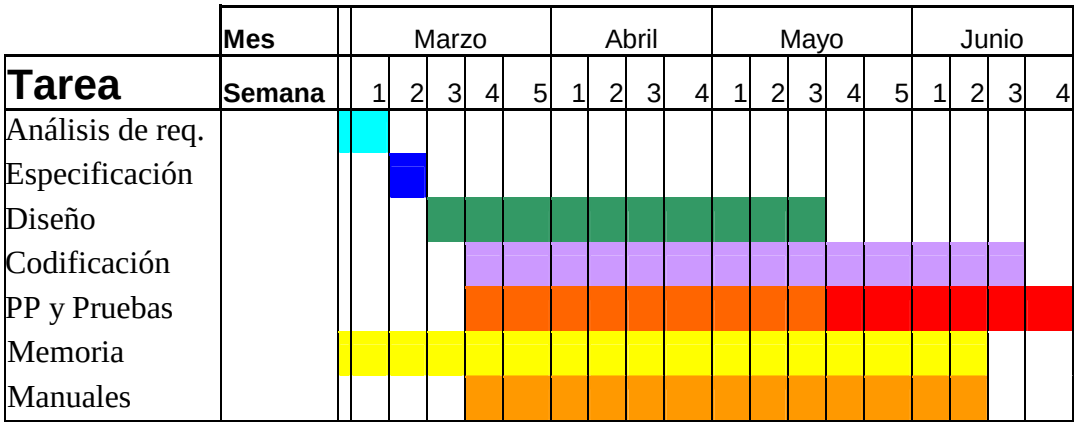
El retraso que se ha producido en la planificación ha venido dado por las dificultades que se han encontrado en algunos puntos del proyecto, por ejemplo en la familiarización con el entorno de trabajo. También se ha producido un retraso considerable con algunos problemas que han surgido durante la implementación, los cuales han sido "pesados" de resolver.

Sin embargo estos retrasos se han ido compensando con otros apartados del proyecto los cuales se han podido realizar antes del tiempo previsto, ambas cosas han permitido que la planificación inicial no se alejará tanto de la planificación final.

A continuación se muestra el esquema de la planificación inicial, cuya medida se realizó en semanas:



Sin embargo la planificación final nos ha quedado de la siguiente manera:



Como podemos comparar los periodos que fueron peor planificados fueron los de Codificación, Puesto a Punto y Pruebas. Los cuales han ocupando un mes más que en la planificación inicial. Hay que destacar que la etapa de pruebas no es que fuese mal planificada, lo sucedido es que tiene una gran dependencia de la etapa de codificación.

En la siguiente tabla se muestra la planificación de forma más detallada, nos permite comparar la planificación final con la inicial y podemos encontrar los tiempos aproximados en horas.

Tarea	Tiempo Previsto	Tiempo Necesitado
Gestiones Administrativas	4 horas	2 horas
Familiarización con el problema y el entorno de trabajo	20 horas	28 horas
Especificación / Análisis de requerimientos	100 horas	70 horas
Diseño gráfico de la aplicación web	20 horas	15 horas
Implementación	250 horas	300 horas
Introducción de datos y pruebas	50 horas	20 horas
Memoria del proyecto	60 horas	50 horas
Total	494 horas	485 horas

(Hay que recordar que el número de horas es aproximado ya que es muy difícil calcular el tiempo dedicado en cada sesión de trabajo a cada etapa del proyecto.)

6.2. Coste económico

La aplicación web resultante no utiliza ningún tipo de software de propietario. Todo el software sobre el que se ejecuta la aplicación es gratuito: PHP, Apache y MySQL.

Por lo que referencia al desarrollo se ha utilizado una combinación de software gratuito y software de pago, exclusivamente Word y Windows, pero estos son ofrecidos por la universidad debido a un acuerdo con Microsoft, así que no hemos tenido que pagar ninguna licencia por ellos. Si se hubiese tenido que pagar licencia, se hubiera recurrido a programas similares del software gratuito como puede ser el Open Office.

A continuación se realiza una estimación del coste de realización del proyecto si éste se hubiese llevado a cabo en una empresa. Para el cálculo del proyecto se han separado los diferentes roles que he tenido que adquirir y los diferentes salarios que se hubiesen recibido a cambio del trabajo realizado.

Los roles que se han tenido que adoptar han sido los siguientes:

- Analista (estructuración)
- Diseñador Web (diseño gráfico)
- Programador (implementación)

A continuación se muestra una tabla con el precio medio de la hora de cada uno de estos puestos de trabajo:

Puesto de trabajo	**Precio medio por hora**
Analista	**15 €**
Diseñador Web	**10 €**
Programador	**13 €**

Si acumulamos las tareas del apartado anterior según el trabajador que las desempeña, obtenemos los costes totales de pago de personal del proyecto:

Puesto de Trabajo	**Tiempo dedicación**	**Sueldo Final**
Analista (Especificación + Familiarización)	98 horas	1470 €
Diseñador Web (Diseño gráfico)	15 horas	150 €
Programador (Implementación + pruebas)	320 horas	4160 €
Total	**433 horas**	**5780 €**

Como podemos observar el coste salarial de los empleados hubiese sido de unos 5780 € aproximadamente. A continuación calcularemos el coste estimado de la maquinaria.

La maquinaria utilizada a lo largo del proyecto ha sido un ordenador de sobremesa y un ordenador portátil. Las características de cada uno de ellos son las siguientes:

Ordenador de sobremesa:

AMD Athlon XP 2700 +
512 Mb de memoria RAM
DVD-ROM
Monitor 17”

El precio aproximado de un equipo de estas características en el mercado ronda los 450 €.

Ordenador portátil:

Intel Core Duo 2 x 1,7 Ghz
1024 Mb de memoria RAM
DVD-ROM
Pantalla de 15,4”

El precio aproximado de un equipo de estas características en el mercado ronda los 700 €

A estos precios calculados hay que añadirle los precios de acceso a Internet más el precio del router de aproximadamente 120 €. Así que el coste final del proyecto lo podemos ver representado en la siguiente tabla:

Tipo de Coste	Coste Aproximado
Sueldo de los trabajadores	5780 €
Adquisición Ordenador Sobremesa	450 €
Adquisición Ordenador Portátil	700 €
Acceso a Internet (4 meses)	120 € (4 x 30 €)
Router	120 €
Total	**7170 €**

Hay que destacar que en el cálculo de este presupuesto no se han contado las máquinas que forman el cluster nozomi2 a las cual accede mediante el CONDOR. Tampoco se ha tenido en cuanta la máquina ba0 en la cual ha quedado instalado el programa. Estas máquinas no se han tenido en cuenta porque el coste ha sido realizado para el proceso de producción y he considerado que éstas no forman de él, ya que forman parte del lugar donde quedará instalada la aplicación.

7. Conclusiones

Este último capitulo tiene tres subapartdos. En primer lugar tenemos un apartado dedicado a los resultado y los objetivos cumplidos a lo largo de la realización del proyecto. Como segundo apartado tenemos un pequeño comentario de posibles mejoras en el proyecto y en su entorno y para finalizar encontramos un comentario personal sobre lo que ha sido para el autor la realización del proyecto.

7.1. Resultados y Objetivos

Todos los objetivos fijados inicialmente han sido cumplidos más o menos en el tiempo previsto. Inicialmente se emulo a nivel local un sistema similar al que actualmente se utiliza para la ejecución de la aplicación. Así que después de descargar de Internet los programas necesarios, éstos fueron instalados en un ordenador personal. A partir de este momento se podían realizar pruebas del proyecto a nivel local, ya que se había instalado Apache, PHP y MySQL.

Una vez instalado el lugar de emulación se decidió definir la estructura visible de la aplicación web, considerando y evaluando las necesidades da cada una de las secciones que iban a ofrecerse en la aplicación.

Simultáneamente se comenzaron a realizar pruebas y pequeños programas para aprender a utilizar el PHP, ya que hasta el momento era un lenguaje desconocido con el que nunca había tenido la oportunidad de trabajar.

Una vez se empezó a dominar el lenguaje de programación se trabajó en la estructura interna de la aplicación. Se creó la estructura de directorios para guardar los ficheros resultantes de las ejecuciones y ficheros internos donde se almacenan funciones en PHP que forman parte de la aplicación, aunque no sean visibles a los usuarios.

A continuación se creó una plantilla web que permitía la incorporación de todas las secciones que se habían pensado para el proyecto, además de la sección de control de usuarios. El código de la plantilla se ha separado en partes independientes para poder dibujar páginas dinámicas con cierta facilidad. Una vez diseñada la plantilla se empezó con la implementación de las funcionalidades de la aplicación, realizando en todo momento pruebas de lo que se iba escribiendo.

El siguiente paso fue la creación del registro de usuarios, y la diferenciación de los diferentes tipos de usuarios. Creando el espacio privado de la web. Por último se incorporaron los ejecutables a la aplicación permitiendo así que todos los usuarios pudiesen ejecutar sus propias instancias en las heurísticas.

El último paso ha consistido en publicar todas las instancias públicas y acabar de introducir los datos en las diferentes secciones de la página web.

7.2. Posibles Mejoras

En este apartado incorporaría que al proyecto se le podrían añadir las ejecuciones paralelas.

También se podría pensar en crear una aplicación encargada de instalar y configurar toda la aplicación dejándola lista para ser utilizada, ya que resulta un poco engorroso el tener que realizar cambios en los ficheros de configuración.

Por último es complicado añadir nuevas heurísticas a la aplicación, así que también podría ser una mejora de la aplicación, facilitar el modo de añadir heurísticas a la aplicación.

Por otro lado propondría al LCLSI que pensasen en realizar un documento o incluso un proyecto basado en el funcionamiento del CONDOR, ya que facilitarían mucho la tarea de los proyectistas debido a que todos tenemos problemas con las ejecuciones de CONDOR.

7.3 Comentario Final

Para acabar con este documento me gustaría escribir un comentario personal sobre lo que ha sido para mi la realización de este proyecto.

La decisión de decidir realizar este proyecto fue una decisión inesperada, ya que inicialmente la idea era proponer un proyecto a algún profesor y que éste lo guiará. Sin embargo, el cuatrimestre anterior no me dejo demasiado tiempo para pensar un proyecto que me motivase, así que comencé a mirar los proyectos ofrecidos por la FIB, entre ellos se encontraba uno que me llamo la atención dirigido por Fatos. Cuando quedé con Fatos me comentó que el proyecto de la web no estaba disponible, pero me ofreció el proyecto que al final he realizado. El proyecto cumplía una de las condiciones que me había puesto a nivel personal para realizar el proyecto, la condición era que el proyecto que realizase no podía quedar sin uso.

A nivel personal el proyecto me ha aportado conocimiento sobre PHP, HTML y incluso he aprendido a montar un servidor y a configurarlo. Así que aunque el proyecto después quede en el olvido, personalmente me he sentido satisfecho al realizarlo.

Me gustaría agradecer a Fatos Xhafa las ideas que ha aportado al proyecto y la dirección que ha realizado del mismo. También agradecer al laboratorio de cálculo del LSI los servicios ofrecidos y la ayuda que han ofrecido durante la realización del proyecto. Tampoco puedo olvidar agradecer a David Martos, otro proyectista, la ayuda que me ha ofrecido durante la implementación del proyecto.

Por último agradecer a toda mi familia y mis amigos los buenos años que me han hecho pasar en la FIB, el apoyo que me han dado tanto en los buenos como en los malos momentos y la ayuda que me han ofrecido tanto en la realización del proyecto como a lo largo de la carrera.

Javier Miranda Jiménez, 13 de junio de 2007

8. Bibliografía

Manuales y información sobre PHP y HTML:

http://htmlhelp.com/es/reference/css/properties.html
http://es.php.net/manual/es/
http://es.tldp.org/Manuales-LuCAS/manual_PHP/manual_PHP/
http://www.webestilo.com/php/php04c.phtml
http://htmlhelp.com/es/reference/css/properties.html
http://www.w3schools.com/html/html_forms.asp
http://msdn2.microsoft.com/en-us/library/ms535261.aspx
http://www.webtaller.com/construccion/lenguajes/php/lecciones/funciones-acceso-archivos-php.php

Consultas relacionadas con el proyecto:

http://www.google.es/
http://es.wikipedia.org/

PFCs de la FIB:

Serveis Computacionals en Internet per l'Optimització Combinatória
Gómez Andía, Raúl.

Algorismes evolutius per al problema de Resource Allocation on computacional Grids Duran Relat, Bernat

Estudi comparatiu d'implementacions de heurístiques per al problema de "Job Scheduling on the Grid
Carretero Casado, Javier Sebastian

Apuntes propios conseguidos a lo largo de toda la carrera:

Apuntes de BD
Apuntes de ES1
Apuntes de ES2
Apuntes de PROP

Estudio sobre el salario del sector informático en España (2002)

http://banners.noticiasdot.com/termometro/boletines/docs/paises/europa/espana/sedisi/2003/sedisi_Resumen_Ejecutivo_de_Salarios_2002.pdf

Coste de los componentes informáticos, extraídos de la tienda PC Green:

http://www.pcgreen.com/

Anexos

En este apartado se han realizado un seguido de manuales que pueden ser de utilidad para la prueba y ejecución del proyecto realizado.

Los manuales realizados son los siguientes:

- **Instalación de un Servidor Local:** consiste en una recopilación de manuales de Internet los cuales se han resumido y se han adaptado a las necesidades de nuestro proyecto. Pensé en realizar este manual mientras instalaba el servidor para realizar las pruebas locales, ya que me surgieron algunos problemas con las configuración de los ficheros de los distintos programas.

- **Manual de Instalación**: Más que un manual, son unas breves instrucciones a seguir para asegurarnos que la aplicación funcionará sin problemas.

- **Manual del Administrador**: En este manual se explica como el administrador debe o puede realizar las funcionalidades que le ofrece la aplicación web.

- **Manual del Usuario:** Esta manual esta dirigido a los Invitados e Investigadores y al igual que el manual del administrador, intenta explicar a los usuarios como utilizar la aplicación web.